A HYBRID APPROACH FOR FAQ RETRIEVAL TASKS

by

Ran Zheng

B.S., Beijing Union University, P.R. China 1998

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

MATHEMATICAL, COMPUTER AND PHYSICAL SCIENCES

(COMPUTER SCIENCE)

THE UNIVERSITY OF NORTHERN BRITISH COLUMBIA

September, 2005

Abstract

FAQ (Frequently Asked Question) collections are constructed to solve users' problems by providing related information in question-answer pairs. The task of searching a large FAQ collection can be viewed as the automatic retrieval of an FAQ entity that semantically matches a user query written in natural language. However, this task remains unsolved due to various difficulties. This thesis presents an approach comprised of three IR methods: the term expansion technique, differential latent semantic indexing (DLSI) approach, and template structure with related algorithms for solving FAQ retrieval tasks. This thesis is based on studying the advantages and drawbacks of traditional *Information Retrieval* (IR) techniques already applied to solving other IR problems, early research works related to retrieving FAQs, as well as the template approach that has been used successfully in language tutoring systems. Finally, by giving detailed methodologies of this hybrid approach for solving FAQ retrieval tasks, the effectiveness of the approach is evaluated through a series of experiments with a particular FAQ collection.

Table of Contents

List of Figures

List of Tables

Acknowledgement

I owe my deepest thanks to Dr. Liang Chen, my supervisor throughout my studies and research at UNBC. Without him, this thesis would not exist. He has given me effective guidance with knowledgeable suggestions on academic research, and provided me with advice on daily life. I feel grateful to him for any future success I may have in this field.

I also would like thank to Dr. Charles Brown and Dr. Jianbing Li for their thoughtful suggestions on this thesis. Their valuable ideas have helped me to complete a few ideas.

I should also thank Dr. N. Tokuda, Dr. A. Nagai and Mr. R. Chen, cooperators of my supervisor, for their early work on this topic which my research is based on, and for their significant contributions to the research this thesis reports

Jia Zeng and Daniel Yule offered many great suggestions on thesis writing. Finally, I especially thank my wife and my parents for unwavering support throughout my years of study.

Chapter I Introduction

This chapter presents the motivation that inspired this research work, an outline of the research procedure and corresponding goals of this research work, as well as the structure of this thesis.

1 Motivation of the Research

Given current widespread usage of the World Wide Web (WWW), FAQs (Frequently Asked Questions) offer people possible ways to solve their problems at low cost. Because of the large amount of problems that users often face in related fields today, most enterprises, governments and international organizations utilize FAQ collections to answer particular questions. However, this kind of organizational form of information has by nature three obvious shortcomings. First, FAQs contain static information seldom modified after generation. Users must choose lexical expressions that are similar as what they want within the contents of the FAQ collection. Hence, fetching valuable information from FAQs highly depends on the users' ability to express what they want in vocabulary appearing in the FAQs. Second, since all FAQs of a collection focus on details of a specific topic, they are generally semantically related. As a result, it is difficult for the users unfamiliar with the general topic of a FAQ collection to distinguish the meaning of each FAQ entry. Hence, extra domain knowledge is required for assisting users to filter FAQs. Third, the number of FAQs in a FAQ collection varies from several to thousands. Manually searching for information in each FAQ

of the collection is very tedious. An automatic searching method is expected for complete and precise retrieval of FAQs which are semantically related to a user's question. Hence, an approach based on Information Retrieval (IR) techniques for assisting users to find one or more correct answers is called for.

Intuitively, by considering acceptable retrieval performance with low complexity in implementation as shown in other IR topics, keyword-based matching approaches should be introduced for retrieving FAQs. However, it is noteworthy that keyword-based matching approaches are less effective at matching user queries to FAQs than matching keywords to documents due to the vocabulary gaps [Sne99].

To avoid this difficulty, a few approaches [BHK+97, Sne99] based on the combinations of Natural Language Processing (NLP) techniques and IR techniques have been introduced to derive the semantic relationships between the words in a user's question and the words in an FAQ. However, these approaches have difficulties in determining semantic relationships among words without good external human references for problems of natural language, such as synonymy and polysemy. Hence, improvements derived from overcoming these drawbacks are still required.

2 Research Procedures and Main Results

Development of an FAQ retrieval system able to accept any free-format user's query written in any ordinary language, such as English, able to return FAQs semantically related to the

user's query in a candidate list without huge human effort is expected.

For the FAQ retrieval task, the study of applying the DLSI (Differential Latent Semantic Indexing) approach [CTN2001] for retrieving FAQs by solving inherently difficult synonymous, polysemous or ambiguity-related problems is the first focus. Term expansion technique is introduced for enhancing the performance of DLSI approach by ensuring queries and results follow similar probability distributions of lexical information.

Second, there is an inherent problem in using the DLSI approach to solve FAQ retrieval: the sequence of retrieval results ordered in the semantic similarity hardly reflects what users really want due to extremely sparse lexical information in user's queries. By assuming that user's queries on the same FAQ are usually constructed in similar patterns, this problem was addressed by appending unseen patterns of various users' questions into corresponding FAQs for further addressing syntactic similarity between a user query and a semantically related FAQ. After this first stage, application of the DLSI method to narrow down searching space for retrieval of semantically related FAQ entities, the second stage was addressing the syntactic similarity with a form-based searching strategy to pin down the FAQs. This was accomplished by adopting a template structure [CT2003] as the mechanism for storing question portions of FAQs in a particular way, and developing a template matching and merging algorithm for determining the proper FAQs in the set of semantically related FAQs returned from the DLSI approach. Furthermore, appending useful patterns of users' queries into the relevant FAQs can be helpful for addressing the syntactic similarity among them.

Consequently, the main contributions of this research are as follows:

- Development of an effective hybrid approach for solving FAQ retrieval tasks by applying three IR approaches: the term expansion technique, DLSI approach, and template structure with its matching and merging algorithms
- Demonstration of the effectiveness and advantages of the hybrid approach through conducting a series of experiments based on the comparisons of different approaches.

3 Structure of this Thesis

In Chapter II, by reviewing background information on related definitions of FAQ and obstructions of retrieving FAQs in detail, a few classic information retrieval techniques and their extended works are studied for their usefulness in solving the FAQ retrieval task. In addition to this, three earlier approaches to solve FAQ retrieval problems are also studied.

In Chapter III, for overcoming particular limitations of retrieving FAQs learned in the pervious chapter, a hybrid approach combining three particular methods is proposed, and methodology details of this new approach are presented.

In Chapter IV, for showing the value of the hybrid approach, a series of experiments based on a specific FAQ collection are conducted to compare performances demonstrated by standard IR techniques and standard versions of those IR techniques combined with the new techniques suggested in this thesis.

Chapter V provides a summary of this thesis and proposals for future work.

Chapter II Background Review

In this chapter, by giving the background information of FAQs, notable obstructions on retrieving FAQs are clarified. Some standard IR approaches and early research works related to retrieve FAQs are also studied.

1 FAQ Collection and FAQ Retrieval Task

A collection of FAQ (Frequently Asked Questions) is a series of questions and corresponding answers that pertain to a certain topic. Each question-answer pair in a FAQ collection is called an FAQ entity and usually extracted from messages posted, for example, to Usenet Newsgroups, in order to reduce repeated answers to similar questions. Currently, the information organized in FAQ form has been widely adopted for solving a series of problems on special subjects at low cost. Thousands of FAQ collections are available on many subjects from various fields. Several sites catalog them and provide search capabilities-- for example, Internet FAQ Archives[1].

FAQs are normally stored in database management systems, or in simple text files. Hence, searching for information in each FAQ entity of a FAQ collection in order to answer users' random questions could be treated as a typical task of IR as retrieving documents aimed to satisfy users' information needs which are usually expressed in natural language [RB99].

[1] http://www.faqs.org

Accordingly, the FAQ retrieval task can be defined as an Information Retrieval task which is designed to meet the informational needs of users by retrieving FAQ entities semantically matched users' questions written in an ordinary language, such as English.

2 The Notable Difficulties in Retrieving FAQs

An FAQ collection provides a set of typical questions and corresponding answers on certain topics. Although users benefit from FAQ entities organized by domain knowledge exporters, three notable difficulties in retrieving FAQs can depress their efficiency in practical applications.

(1) A traditional FAQ collection serves users in a passive way.

Users never have any chance to retrieve answers by asking questions in their own words. Instead, they must try to discover related FAQ entities through applying human intelligence to study meaning connections one by one. Since manually scanning the whole FAQ collection and matching entities and human users' requests is tedious, and often users cannot discover related FAQ entities even when the FAQ collection contains exactly what they want.

(2) A domain knowledge gap obviously exists between FAQ collection creators and users.

Since the lexical information of questions asked by a particular user are always different from typical questions originally contained in FAQs, users apply familiar domain knowledge along with what they learn from the question-answer pairs contained in the corresponding FAQ collection to keep rephrasing their questions and to guide them until eventually stumbling on

the correct FAQ. However, the experts and creators of FAQ collections always compose FAQ entities with a great deal of domain-related words or technical terms. As a result, it is hard for users lacking similar background knowledge to find what they want represented in a set of technical terms.

(3) FAQs may be poorly organized within a collection.

FAQs are usually summarized and grouped in a particular order within a collection by different people. Since the basis for understanding FAQs may vary from person to person, information contained in FAQ entities is rarely organized in a unified standard. Hence, for users, it is unlikely to find related FAQ entities based on prearranged categories within an FAQ collection.

According to these difficulties of retrieving proper FAQs stated above, approaches designed for helping users overcome difficulties are expected.

3 Related IR Techniques Considered in Solving FAQ Retrieval Task

By defining the FAQ retrieval task as a topic of IR, intuitively, the IR techniques applied in other IR topics can potentially be applied to retrieving FAQ entities related to any given users' queries. Following, some familiar approaches of IR are studied by decomposing the FAQ entities and users' queries into a group of words.

3.1 Classic Boolean Approach and Extended Boolean Approach

The widely used classical Boolean approach is introduced in Cooper's 1988 work [COO88].

The relevance of any information entities, such as documents, and the users' queries represented in Boolean expressions is judged by classical set theory. For example, the given query, "*t1* and *t2*" matches documents which contain the terms *t1* and *t2* at the same time. Hence, semantic similarity of any two information entities can be evaluated by using proper Boolean operators to establish the relationships of terms appearing in those information entities. Methods based on the classical Boolean approach are widely applied in commercial applications, such as web searching engines, and provide acceptable retrieval performance with low maintenance costs. Since there are only two possible results from Boolean logic, *true* or *false*, people find that an efficient Boolean expression for representing complicated relationships among the words is hard to define.

To overcome this, extended Boolean approaches introduce the term-weighting [Bue81, BOO80, SFW83] method and soft Boolean operators (also called fuzzy Boolean operators) [LKL92, LKK+93]. By giving a weight between 0 and 1 to each term instead of only 0 or 1, the characteristics of a document can be described by the weight of each term in the document. Through fuzzy Boolean operators, documents can be evaluated by similarity to user queries. By employing the term-weighting method with soft Boolean operators along with the classical Boolean approach, this extended Boolean approach demonstrates good ability to evaluate relevance between documents and a user queries [Lee94, SFW83].

By following classical or extended Boolean approaches, relationships among terms in each FAQ entity or user's query can also be directly interpreted by translating their contents in

Boolean expressions. However, [VVW+97], neither classical nor extended Boolean approaches are sufficient for interpreting the semantic similarity between FAQ entities and users' queries due to using limited vocabulary to represent boundless meanings in FAQ entities and the users' queries.

3.2 Vector Space Model (VSM)

To avoid the difficulty in using Boolean operators to represent complicated relationships among words in documents, the Vector Space Model (VSM) [SMM83] introduces an innovative way to represent the content of a document as vector based on the occurrences of words or terms appearing in the document. VSM approaches are very successful in solving IR problems through applying the theory of linear algebra to discover the relationship between the vectors of any two documents instead of evaluating the similarity of any two documents by interpreting the meaning of the documents directly.

VSM approaches usually consist of three major steps: index valuable words that appear in a document; construct vectors for representing unique characteristics of each document by weighting the valuable words which appear in these documents; rank relevant documents with respect to relationships between the vectors of users' queries and each document in a document collection as evaluated by applying the theory of linear algebra.

The first step involves indexing terms which are specific words within documents and defined by case oriented principles. Different terms contained in a particular document help

people to distinguish the document itself from others in a document collection. To index terms of documents more accurately, two standard techniques – stop word list and word stemming – are usually applied in VSM. In natural languages, such as English, a stop word list is a collection of stop words [vR79], which appear frequently in the document text and have little impact on its content. Examples of stop words in English are articles, prepositions and conjunctions, such as "the", "a", "an", "this", "that" and so on. Another standard technique for indexing terms efficiently is word stemming. For IR purposes, people [Pro80] usually thought morphological variations of words that, in most cases, have equivalent interpretations. Thus, anyone of word stemming methods, such as Porter Stemmer [Por80], aims to follow particular rules to reduce words to their grammatical roots for avoiding this problem. By applying these two standard techniques, useless words and affix of words are eliminated. As a result, the content of any documents can be represented precisely by relatively small number of terms in unified forms and requires much lower storage space.

The second stage consists of giving a weight to each term to represent its importance to a document within the document collection. In this way, any document can be explicitly distinguished from others by addressing the difference among statistical distributions of terms in different documents. Normally, when weighting terms, two factors are concerned by researchers [Fra92][Sal73]: the appearing frequency of each term within a single document, describing the relationship between each single term and different documents, and appearing frequency of terms within the whole collection, describing the relationship between each

single term and a document collection [Fra92]. Among the weighting strategies based on these two factors, two important methods are reviewed here. One is the *tf-idf* weighting scheme [Sal73] which multiplies each term's *tf* factor by its *idf* factor in each document. The *tf* factor is a normalized measure of how important the term is for that document. It is calculated by dividing a term's frequency in the document by the maximum frequency of any term in the same document. The *idf* factor is defined as inverse the document frequency, and calculated by the total number of documents in the collection divided by the number of documents containing a particular word. This factor is designed to make rare words more important than common words. The point of this strategy is to balance term weights by the two factors for avoiding overemphasis or de-emphasis either frequently or rarely appearing words. As the researchers have noticed, a term that appears equally in all the documents is much less important for characterizing a document than a term that appears concentrated in only a few documents. The *log-entropy* scheme [Dum92] aims to weaken the effects of terms occurring frequently or distributed equally across the documents in a document collection by giving them less weight, and to highlight the terms that can distinguish a document from others by granting them much more weight. Like the *tf-idf* approach, the *log-entropy* scheme is balanced by the logarithm of term frequency, a factor reflecting the importance of the term for each single document, and the information entropy of the term, a factor representing how much information the term carries across all the documents of a collection. The *log-entropy* weighting method is widely applied in the LSI (Latent Semantic Indexing) and its enhanced DLSI (Differential Latent Semantic Indexing) approaches. Hence, the meaning of any

document within the collection could be represented as a vector, also called as a document vector. In VSM, all document vectors of a particular collection are constructed in a unified form. Each dimension of a document vector represents a term and is assigned by a weight of the term in a particular document.

In the third stage, a vector space, also called as a term-document vector space or a content space, is constructed by the document vectors of all documents in a collection. Consequently, the problem of evaluating the similarity of any two documents is changed into discovering relationships between two vectors in a vector space by associative coefficients. For measuring the similarity of any two document vectors, several methods, such as cosine measurement [Sa188], have been developed. As a popular similarity measurement, the cosine coefficient measures the angle between any two given document vectors in a corresponding vector space. A smaller angle between two vectors represents a stronger relevance between those two documents.

By following the three steps described above, VSM approaches offer a much easier way to interpret the meaning of documents by considering the statistical distribution of terms in each single document and in a corresponding document collection. VSM provides an impressive way to retrieve related documents by evaluating the semantic similarity between a user's query and each document based on automatic analysis which does not require external references for interpreting the relationships of any two documents. Hence, by treating users' queries and FAQ entities as documents, the FAQ retrieval task could be solved by approaches

based on VSM. However, traditional VSM assumes that terms, the coordinates of the space, are all "meaning" independent of each other, thus problems in processing natural languages, such as the polysemy and synonymy, are ignored. Hence, it is hard to discover the relationships of same terms with different meanings, or different terms with the same meaning [Fra92].

3.3 Probabilistic Approach

As Cooper [COO88] thought, probabilistic approaches use probabilistic design methodology to estimate the probability of relevance of any two documents rather than the Vector Space Model that applies simple similarity measures (e.g. the cosine measurement). The probabilistic approach normally calculates the "conditional" probability $P(D|R)$ that the relevance of a given document D and particular condition R is viewed on a random basis [Sal89]. For solving different cases of IR tasks, conditional probability is calculated by specific models and hypotheses on the distribution of terms, such as Bayesian probability models [TC90] and the Binary independence model[vR79] [Eft95]. By doing so, probabilistic approaches demonstrate a remarkable ability to infer relevance of given queries and documents by applying properly probability theory and statistical information to address independent evidence.

As shown in the probabilistic approaches for solving IR problems, there are two obvious problems. First, term mismatch [JXu97] is a fundamental problem in IR topics. It is difficult for users to apply the terms within related documents to compose queries since vocabulary

may greatly differ from person to person. Second, because of the inherent ambiguity of natural language, the meaning of a word used by a user highly depends on her/his understanding of the context. As a result, it is sometimes extremely difficult, if not impossible, to understand the exact meaning of a word in a user's query without other references. To avoid these two problems, methods [Har92] [BAS94] based on the query expansion technique are introduced. Query expansion is a technique for learning the co-relationship between any two terms described by the conditional probability $P(D|R)$ [CWN+02]. Query expansion increases the possibility of retrieving correctly-matched "answers" by appending additional terms into user queries directly based on particular probabilistic estimation formulas [FLG+87].

As a problem of the FAQ retrieval task, knowledge gap between users and FAQs creators exists. The query expansion method is most interesting due to its ability to bridge huge lexical variations between users' queries and FAQ entities and for improving the probability of retrieving correctly-matched answers. For the query expansion method, the decision of appending particular terms into users' queries is only made by reviewing relevance feedbacks given by users. The performance of the approach involved with this method highly depends on quality of feedbacks on particular FAQs. Furthermore, for the term expansion technique, probabilistic approaches are not widely used, because efficiency of probabilistic approaches highly relies on applying a specific probabilistic model for a particular IR problem. Hence, to avoid the problems of the term expansion technique, necessary modifications for adapting

FAQ retrieval tasks are expected.

3.4 Latent Semantic Indexing (LSI) and Differential Semantic Indexing (DLSI)

By assuming each dimension (term) of the semantic similarity analysis space is "meaning" independent of each other in each document and in a document collection, approaches based on traditional VSM for solving IR problems usually only consider term-document relationships. However, term-term relationships, such as synonymous words appearing in a similar context, ought to be addressed at the same time. Thus, the LSI method [DDF+90] and its enhanced approach, DLSI method [CTN01], have introduced innovational ways to explore semantic relevance by reviewing term-document relationships as well as the term-term relationships within the collection.

(1) LSI approach

Inheriting the capability of approaches based on traditional VSM in addressing term-document relationships, Latent Semantic Indexing (LSI) attempts to explore the relationships among terms, such as synonymous and polysemous words, by addressing the conceptual terms which are truly meaning independent of each other[DDF+90]. In the LSI method, term-document vector space, the space originally defined in VSM, is replaced by a lower dimensional document space, called *k*-space (or LSI-space), in which each dimension represents a conceptual "feature" ("factor") which is truly meaning independent of each other and deduced by dimensionality reduction methods. By representing the documents as LSI

factor vectors in *k*-space, the similarity between any two LSI factors vectors in *k*-space can be calculated in the same way as the relevance of vectors in traditional term-document vector space can be evaluated. Bartell believes documents and queries dealing with the same topic would be far apart in the traditional term-vector document space because they use different but synonymous terms, but would be close together in the *k*-space [BCB92] because the relationships among semantically related words are captured through statistical analysis of terms appearing in the document collection without any extra natural language analysis [Hul94].

By using Singular Value Decomposition (SVD) to discover such a lower dimensional space, LSI takes the original term-document matrix, *X*, which is *m* by *n*, where traditionally *m* is the number of terms and *n* is the number of documents, as the input. It puts out three new matrices: *T*, *S*, and *D*, where $X = T \times S \times D$. The matrices *T* and *D* are called left and right singular vector matrices correspondingly, and consist of orthonormal columns. The matrix *S* is a diagonal matrix of singular values of the original matrix *X* arranged in descending order. By simply keeping the largest *k* ($k>0$) singular values of matrix *S*, the original high dimensional term-document vector space could be approximated by LSI-space (*k*-space). In the meantime, the left singular vectors matrix *T* is truncated into a $m \times k$ matrix which can be understood as the representation of original terms in *k*-space; the right singular vectors matrix *D* is cut as a $k \times n$ matrix which is interpreted as the representation of original documents in the reduced *k* space. Hence, by decomposing the matrix, *X*, SVD provides a

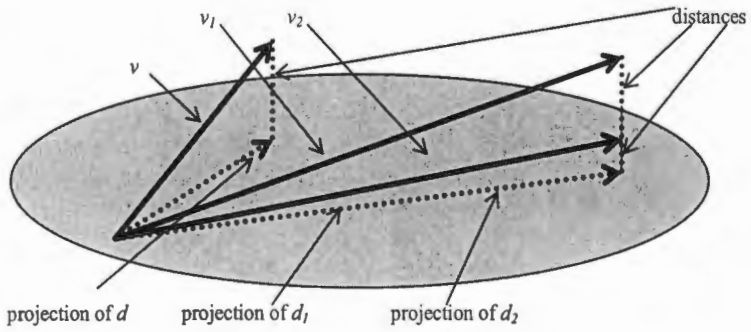
simple way to approximate the original document space in a low dimensional space while keeping most of information presented in the original document space, say, $X \approx X_k$, where each dimension is truly orthogonal in meaning. Similarity between a query and any document in LSI-space is usually calculated by similarity evaluation methods used in traditional VSM, such as cosine measure.

As a very successful method in solving document retrieval problems, the LSI approach is an efficient way to discover the relevance among given queries and documents in a collection without human intervention, even if they do not share exactly the same terms.

(2) DLSI Approach

The traditional LSI method intends to measure the similarity of any two documents through discovering the cosine angle of the projections of any two document vectors within the LSI space. However, researchers noticed [SS97], the LSI approach is a kind of global dimensionality reduction approach which faces difficulty when exploring the unique characteristics of each document by considering the angle of projections of two document vectors on the LSI space only. Thus, by representing the unique characteristics of documents in a particular way, the Differential Latent Semantic Indexing approach [CTN01] considers the following together for evaluating relevance among documents: projection of a vector on a lower dimension space and distance of the vector to the space. As shown in Figure 1, problems, such as distinguishing two vectors that have the same projections on the space and different distances to the space, can be solved by more precise similarity measurements of the

DLSI approach involving two factors, projection and distance, rather than the LSI approach which uses cosine measurement to evaluate similarity



v, v_1 and v_2 are the term vectors of document d, d_1 and d_2. Which document is much more related to document d, the document, d_1, or the document, d_2?

Figure 1: DLSI Approach

In the DLSI approach, unique characteristics of documents are reflected by the interior difference of a document itself and exterior differences between any two documents. Focusing on the different vocabulary used in two different representations of the same document and the different usages of the same vocabulary in different representations, the interior difference of a document emphasizes different characteristics of the document described by two independent representations. Here, the two independent representations of a document describe the whole content of the same document from different viewpoints, such as different summaries of the same document written by different people. Likewise, the exterior difference assesses lexical differences described in the representations of any two documents. By representing a document in a term-document vector in the LSI approach, both interior and exterior differences can be calculated from term-document vectors and arranged

as interior differential vectors and exterior differential vectors, respectively. By constructing the differential term-document vectors, two term-document vectors spaces, interior differential space and exterior differential space, can be established for semantic analysis.

As in the LSI approach, the DLSI approach measures relationships among terms on two lower dimensional spaces, the *k-interior* differential space and *k-exterior* differential space, the dimensions of which represent the conceptual terms as in the LSI approach and are truly meaning independent of each other. By assuming the differential document vectors are constructed in a high dimensional Gaussian distribution [CTN01], the probabilities of any two documents in the *k-interior* differential space and *k-exterior* differential space being the same can be evaluated by two factors: the distance from a differential vector to the reduced differential space, and the length of projection of the differential vector on the reduced differential space. Finally, for combining the probabilities of two documents that are similar to each other but evaluated on the two differential vector spaces, the similarity of a user query and each document in the collection can be interpreted by the posteriori probability function.

By evaluating the semantic similarity between any two documents based on a more precise model, the DLSI approach performs better than the original LSI approach in several IR respects, such as document retrieval [CTN01] and document classification [CTN03]. Hence, the FAQ retrieval task can potentially be solved by using the DLSI approach due to two advantages of the approach. First, the differences between any two independent

representations of one document and between the representations of any two documents can be used to discover characteristics of a document with the lexical information contained in the two representations. Second, a probabilistic model may be applied to evaluate the semantic similarity between a user's query and a FAQ entity more precisely based on two different term-document spaces.

4 Related Approaches to Solving FAQ Retrieval Task

Unlike problems of Artificial Intelligence (AI) which focus on dynamically generating new answers for corresponding queries by analyzing related evidence, the FAQ retrieval task aims to retrieve static answers, an FAQ entity, from a collection of FAQ entities, by evaluating the relevance of given queries and FAQ entities. Several FAQ retrieval systems, including FAQ Finder [BHK+97], Auto-FAQ [Whi95], and EKD QA system [Sne99], for performing FAQ retrieval tasks have been published.

4.1 FAQ Finder

FAQ Finder is a natural language question-answering system that uses files of frequently asked questions (FAQs) as its knowledge base. FAQ Finder applies a hybrid approach of statistical and natural language processing techniques to match users' questions against known FAQ entities from FAQ files, each of which contains a question and corresponding answer pair.

As the first step of this system, FAQ Finder narrows down the candidate FAQ files by using

the SMART information retrieval system [Buc85], an approach based on a Vector Space Model with a precise term weighting scheme, to pick up any FAQ entities possibly matching given queries. As the second step, the FAQ Finder performs a shallow lexical semantics analysis as a more precise matching process. This process is designed for choosing candidate answers highly related to users' input by constructing a simple parse tree and applying an external reference, WordNet. WordNet, a semantic network of English words, takes the role of knowledge provider to reveal semantic relationships between given words and "synonym sets", and among the sets themselves.

By comparing the results of experiments using different approaches, such as applying the SMART system only, and applying the FAQ Finder system with different similarity score functions, FAQ Finder demonstrates a superior performance [BHK+97] on the FAQ corpus. This corpus was obtained from the FAQ collections of the UseNet owned by MIT with the user query set extracted from log files of local users.

Being the first information retrieval system for the FAQ retrieval task, at the first stage, FAQ Finder uses an efficient IR system, SMART, to choose semantic-related candidates. Furthermore, at the second stage, FAQ Finder successfully addresses the semantic relationships of retrieval candidates and users' queries by discovering particular semantic relationships of terms in a simple natural language processing method. FAQ finder is a good example of using a hybrid approach to solve the FAQ retrieval task. However, Burke notes [BHK+97] WordNet is still incompletely constructed due to many missing synonyms, thus

the incompleteness of WordNet lowers the practical performance of FAQ Finder. Hence, Burke suggests an automatic compiling method should be explored for building the appropriate synonym and hyponym links.

4.2 Auto-FAQ

Whitehead [Whi95] states that Auto-FAQ is a text retrieval system not based on the traditional natural language processing system. The Auto-FAQ system distinguishes itself from other question-answering systems in the way it implements three features: query processing, information acquisition, and information management. By focusing on retrieving FAQ entities and query processing, Auto-FAQ applies a so-called shallow language understanding technique in a simple two-stage method. This two-stage method is based on a keyword matching strategy to address the similarity between any users' queries written in natural language and any FAQ entities based on a dense built-in informational database. For processing an FAQ retrieval task, at the first stage, Auto-FAQ applies a simple keyword matching method to compute the matching score between user input query and the question of each FAQ entity in the collection. At the second stage, this system modifies the score of each FAQ entity to magnify in a particular way any scores of entities with high value obtained in the first stage.

Although detailed information of the methods applied in Auto-FAQ have not been released in related references, Auto-FAQ still shows its value for solving FAQ retrieval tasks in three properties of the system according to the experimental results given in the paper [Whi95].

First, and although the so-called shallow language understanding method as a keyword-based approach involves no deduction or inference, this technique still makes the whole system highly efficient in responding time and acceptable performance. Second, Auto-FAQ employs a gap list for storing users' questions for which the system could not find a suitable answer. Although this gap list is maintained off-line by domain experts who manually answer such questions, this idea still contributes to avoiding extreme dependence on the efforts of a small number of knowledge engineers. Last, Auto-FAQ provides an opportunity for users to rate the usefulness of the answers, which can then be used for filtering the unrelated FAQ entities by regarding the predefined threshold. However, because the scoring rules applied in the keyword-based matching method of Auto-FAQ system have to be rebuilt for adapting particular FAQ collections, it requires a great deal of labor to analyze the relationships of keywords and maintain the gap list.

4.3 EKD QA System

As another hybrid approach, Sneiders [Sne99] introduced the EKD (Enterprise Knowledge Development) QA System for solving the FAQ retrieval problem. The EKD QA system organizes FAQ entities based on the Enterprise Modeling technique and applies a more precise keyword-based method, prioritized keyword matching, to better explore the relationships between given queries and FAQ entities. As a major part of this system, the Enterprise Modeling technique takes a role in representing users' queries by applying knowledge contained in the FAQ collection. Furthermore, there is an assumption that each

sentence written in natural language contains three main types of words: required keywords, optional keywords and irrelevant keywords. According to this assumption, prioritized keyword matching implemented by a semi-formal algorithm intends to match keywords contained in any two sentences in the order of the importance of keyword types.

First, the processing procedure of the EKD QA system builds up a knowledge base of a particular FAQ collection by applying the Enterprise Modeling technique to sort and group the knowledge held by that FAQ collection. The questions of FAQ entities in the collection are decomposed into a pattern that consists of primary required keywords, secondary required keywords and other keywords, such as optional keywords and irrelevant keywords. Second, user inputs are decomposed in the same way when the system receives user queries. Third, the system returns a related FAQ entity, or rejects the user, based on computations by the prioritized keyword matching method. Finally, if the user could find the related FAQ entities retrieved by the system, the pattern of the user query can be merged by the system into the pattern of the corresponding FAQ entity. If not, the user is offered a way to send the query back to the FAQ collection maintainers to use human intelligence to analyze relevance between it and FAQ entities in the collection and then manually append this pattern of user query into related FAQ entities.

By interpreting the content of users' queries and FAQ entities based on the particular domain of knowledge held by the FAQ collection, the EKD QA system represents the users' queries and FAQ entities in particular patterns for finding similarities between users' queries and

FAQ entities by matching words with the same word types. There are two ideas shown in the EKD QA system. First, the EKD QA system translates complicate structures of sentences into simple representations by considering the importance of the words in corresponding sentences. By doing so, the system shows the value of using a specific form constructed in particular principles to understand what users input. Second, the retrieval performance of the system can be improved by expanding these patterns of FAQ entities based on studying user feedback. However, the EKD QA system suffers from the same problem as Auto-FAQ. The system requires huge amounts of labor to set up the knowledge base and construct related patterns in order to interpret relationships among terms presented in new FAQ collections.

5 Summary

FAQ collection provides a way for users to solve particular problems through manually or automatically matching their questions with particular FAQ entities created by domain knowledge experts. However, the efficiency of using FAQ collection to solve typical problems on particular topics can be lowered dramatically due to the natural difficulties of retrieving FAQ entities. Considering that an FAQ retrieval task is a problem of IR, several IR techniques that have been applied successfully in solving other IR topics are introduced as potential solutions. Although all approaches contribute to the FAQ retrieval task in particular ways, there are still various limitations to different approaches in retrieving FAQ entities. Likewise, three early approaches based mainly on IR techniques, the FAQ-Finder, Auto-FAQ, and the EKD QA System, were reviewed for discovering achievements and limitations of

practically solving the FAQ retrieval task. As the most common achievement, each of these implementations combine different methods to retrieve related FAQ entities, such as the SMART system with WordNet and the EKD model with prioritized keyword matching. However, as a common problem, the performance of those approaches depends heavily on quality of the knowledge base, such as a complete dictionary of synonymous terms and dense information for illustrating all possible combinations of terms.

Consequently, by considering studies on related general IR techniques and particular approaches presented above, a few core features of an expected approach for FAQ retrieval task warrant consideration.

- Retrieving related FAQ entities should be based on users' queries written in format-free natural language and without particular query language form requirements.
- Since all IR techniques stated above have difficulties retrieving FAQs, a hybrid approach achieved by combining specific techniques can definitely contribute to solving those difficulties.
- The semantic analysis of this new approach should be conducted automatically and without enormous human effort for constructing related references on particular FAQ collections.
- Human intelligence should be considered and applied in a simple way which does not require massive human effort or complicated inference procedures

Chapter III The Hybrid Approach

This chapter describes the architecture and methodologies of a new hybrid approach after considering related IR techniques and early works for solving the FAQ retrieval task.

1 Principles Driving the Design of the System

As show in related research works [CTN01, CTN03], the DLSI approach in solving other IR topics performs well. By applying a particular document representation method, the informational organization form of any FAQ entity naturally consisting of a question and an answer strongly suggests the DLSI approach as the core technique for performing FAQ retrieval tasks. As stated in section 3.4 of Chapter II, in the DLSI approach, two independent spaces, interior differential space and exterior differential space, are established for semantic analysis by using the interior and exterior differential vectors, which represent the differences in a document itself and within any two documents. Hence, the quality of representing characteristics of a document based on content differences depends highly on the differences of vocabulary used in the two representations. Different lexical information from the same document can assist people in understanding meaning. When documents do not contain separate representations for meanings in the natural forms, people must use additional tools, such as auto summarization tools or human intelligence, to create independent representations with different vocabulary for the same document. As the natural form of FAQ entities, two portions, a question and a corresponding answer, are contained in each FAQ

entity as two independent representations in various vocabularies. As a result, one could easily and clearly apply the DLSI approach for solving the FAQ retrieval method by establishing the interior and exterior differential spaces for analyzing semantic similarity of each user's query and each FAQ entity without extra works for generating independent representations of each FAQ entity in a collection.

The efficiency of the DLSI approach heavily depends on whether the probability distribution of terms used in users' queries and in documents are similar. Two things in each FAQ entity and user query can lower the efficiency of applying DLSI to perform the FAQ retrieval task. First, the question portion, entered by users, normally contains terms which rarely appear in the answer portions and vice versa. Examples of these terms include "when, where, how, who, what" in the question portions, and "because, since, hence, therefore" in answer portions. Second, users tend to use simple sentences with few words to express what they want in the question portion shown in each FAQ entity. This lexical information gap directly leads to a huge difference in lexical distribution between the question portion and answer portion of an FAQ entity, or between a user query and FAQ entity. Hence, a method for automatically bridging the lexical distributions of the two portions of each FAQ entity itself, as well as the distributions of given queries and FAQ entities, is needed.

Distribution gaps between users' queries and document collections can be bridged by appending additional terms into corresponding documents. Hence, term expansion has been developed for bridging gaps of lexical distributions existing in the two portions of each FAQ

entity and the gaps between any users' queries and FAQ entities. The term expansion technique considers the knowledge originally held by a FAQ collection other than the query expansion technique, a similar technique, expanding user queries only based on the external evidences, such as users' response.

Usually, users expect to find what they want at the top of a candidate list containing a few FAQ entities related to their particular queries. Although the distinct advantage of the DLSI method in text searching lies in its capability to solve inherently difficult synonymous, polysemous or ambiguity-related problems without resorting to thesaurus or ontology-related vocabulary dictionaries, it returns a candidate list from which users must try to identify what they really want. Even when the list is brief, users can still spend considerable time reading candidates one by one. Although the DLSI method captures similar meanings of documents by abstracting words or terms used in users' queries and FAQ entities, even if very few common terms or words have been used in queries or documents, extra syntactic information, such as the order of common words, held by FAQ entities could help users narrow down semantically related documents at a syntactic level.

A template automation structure originally developed for a language tutoring system [CT03] can play an important role in this issue. This template structure is used to embed many different patterns of semantically similar expressions on particular documents. By employing this template structure for representing users' queries and documents, any syntactic similarity between queries and documents could be discovered by a template matching algorithm.

Hence, for the FAQ retrieval task, by giving a limited number of candidates semantically related to a user query, users' wishes could be pinpointed by evaluating additional syntactic similarities among users' queries and the candidates. Since the structure of a template can contain different representations of the same meaning, the template has the potential ability to improve the precision of exploring syntactic similarity between any two templates by appending various semantically related representations. Therefore, the precision of syntactic similarity analysis can be improved gradually by merging users' queries into the corresponding templates.

As a result, a new hybrid approach for solving the FAQ retrieval task by applying DLSI as the core technique with a few additional methods to dampen adverse effects of the limitations of the DLSI approach could generally be implemented by following the three-step framework shown in Figure 2:

Step 1: Bridge lexical distribution chasms among FAQ entities themselves and distribution chasms between given users' queries and FAQ entities in an FAQ collection by using a term expansion technique to automatically add additional terms into corresponding portions or users' queries.

Step 2: Apply the DLSI approach to retrieve a group of semantically related candidates by solving many of the natural language-related disambiguation problems of contents.

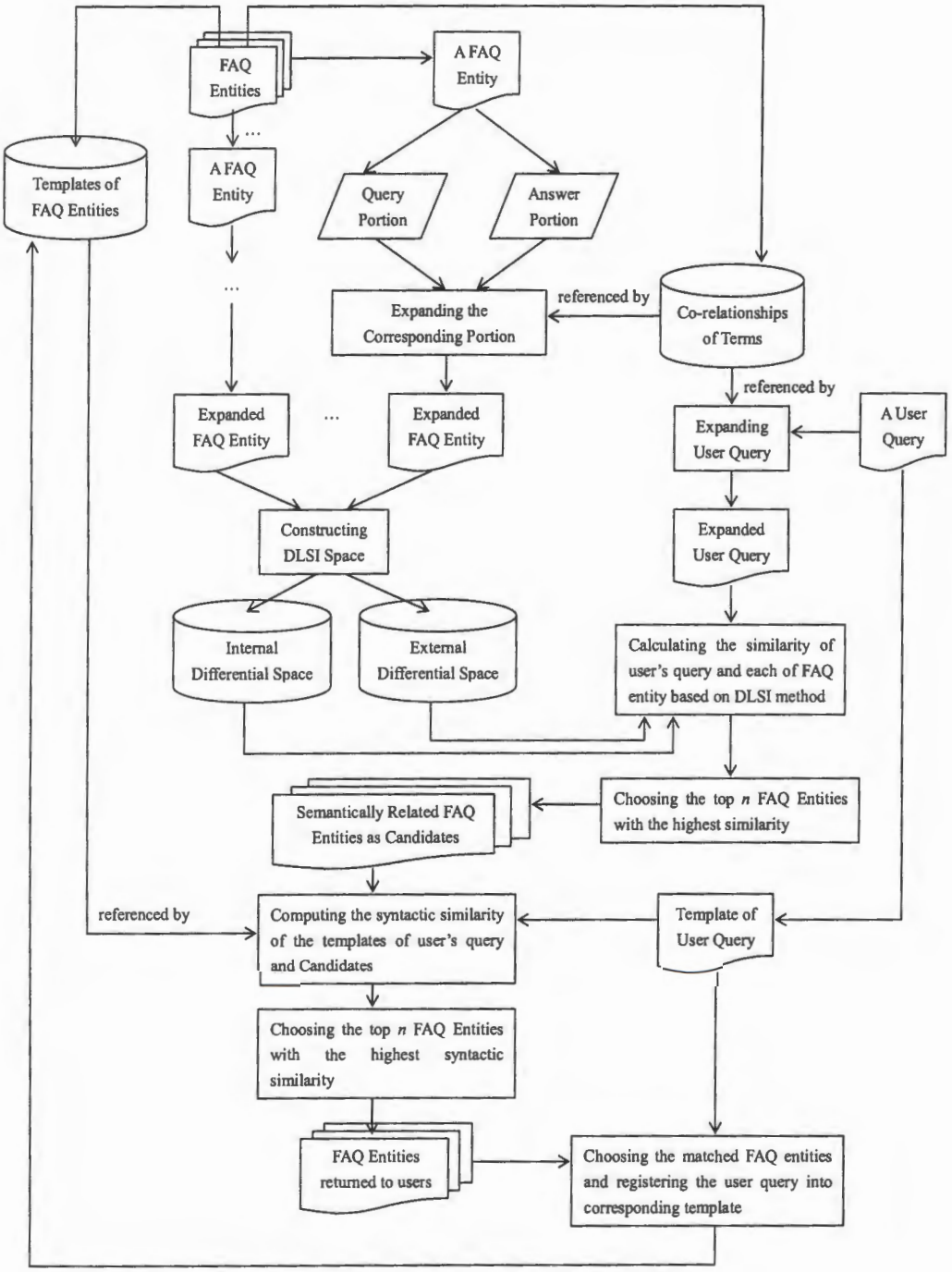


Figure 2: The Framework of the Hybrid Approach

Step 3: Represent users' queries and FAQ entities in a template structure and use a related matching algorithm to address the syntactic similarity between a particular user query and each candidate in a candidate list obtained by applying the DLSI approach. Moreover, the ability of templates to address syntactic similarity can be enhanced by appending various semantically related representations, such as users' queries on particular FAQ entities, into the corresponding FAQ templates.

Consequently, as an IR approach applying two standard techniques, eliminating stop-words and word stemming to index terms[1], the hybrid approach aims to retrieve semantically and syntactically related FAQ entities by regarding a free-format user query written in natural language.

2 Term Expansion

2.1 Basic Strategy

An FAQ collection contains much richer lexical information than each single FAQ entity has, as well as provides the references of different meanings of same terms and similar meanings of different terms in various contexts across an FAQ collection. Hence, by learning the relationships among the terms presented in the FAQ collection, appending additional terms semantically related to the particular contents of original FAQ entities into a particular FAQ

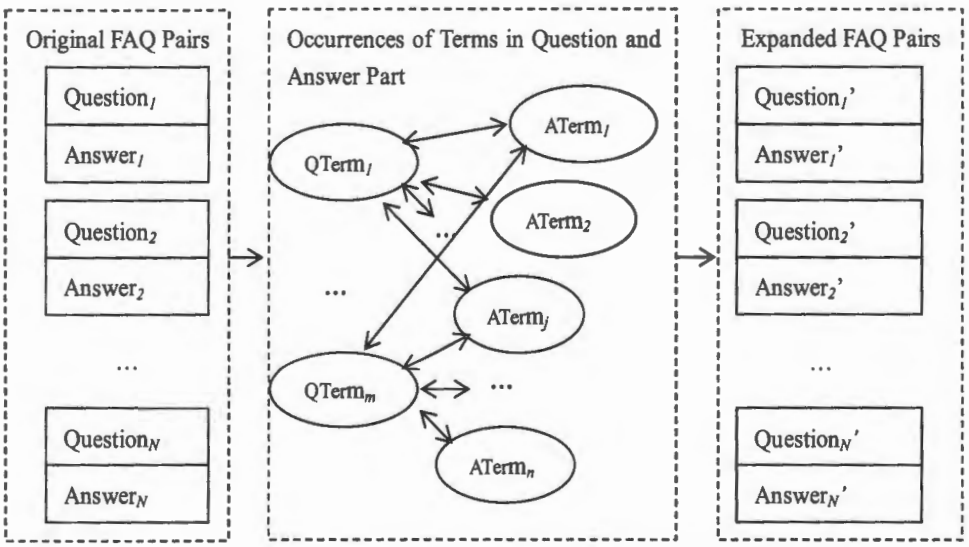
[1] See the detailed information about eliminating stop-word and word stemming techniques in section 3.2, Chapter II

entity could contribute to representing the meaning of an FAQ entity with much richer lexical information.

Several methods [SB90, Sal89, vR79] can be used to pick up additional terms for enriching lexical information in upcoming users' queries by learning relationships among these terms from external references, such as users' feedback or WordNet. Whether terms for expanding queries are selected interactively or automatically, existing methods [XC98, Eft95] normally only consider terms occurring in the relevant document and occurring in non-relevant documents. As a result, before one can decide which terms should be appended into an upcoming user query, proper rules for manually or automatically distinguishing relevant and non-relevant documents are required. Since all FAQ entities in an FAQ collection are somewhat related, it is difficult to set proper rules for automatically judging whether a particular document relates to a user's particular query.

According to the nature of vector space model, initially, terms can be assumed as appearing independently within any portion of any FAQ entity. In order to avoid difficulties in learning such relations among terms based on external references, as shown in Figure 3, a simple method based on the co-occurrence possibility of each two-term pair, one term coming from the question portion and the another coming from answer portion, and the co-occurrence possibility of a term and a group of words contained separately in two portions of the same FAQ entity is introduced for deciding which terms ought to be appended into the corresponding portions. The co-occurrence possibility of a two-term pair, (t_i,t_j), is the

possibility of t_i appearing in a portion caused by t_j appearing in the another portion of the same FAQ entity. The co-occurrence possibility of a term, T_i, and a group of terms, $s(t_1,t_2,...,t_n)$, describes the chance of T_i appearing in a portion caused by any term of $s(t_1,t_2,...,t_n)$ contained in another portion of the same FAQ entity. The co-occurrence possibilities of two-term pairs directly indicate the logical connection of any two terms appearing separately in question portion and answer portion of any FAQ entity. Similarly, the co-occurrence possibility of a term and a group of words gives people a hint about the connections of semantic meaning and logical relationship between a term of a portion and content of another portion of the same FAQ entity.



$QTerm_i$ is a term appeared in question portions of all FAQ entities in the collection.
$ATerm_j$ is a term appeared in answer portions of all FAQ entities in the collection.

Figure 3: Term Expansion

The co-occurrence possibility, $p_{a,\{q_1,q_2,...,q_n\}}$, of a term, a, appearing in an answer portion caused by any word of the word group, $q_1, q_2, ..., q_n$, contained in a question portion of the same FAQ entity could be estimated by applying the following equations:

$$p(a \mid q_i)_{i=1,...,n} = \frac{n(a,q_i)}{n(q_i)} \quad (1)$$

$$p_{a,\{q_1,q_2,...,q_n\}} = 1 - \prod_{i=1}^{n}(1 - p(a \mid q_i)) \quad (2)$$

Where: $p(a|q_i)$ denotes the co-occurrence possibility of a term a appearing in an answer portion caused by the term q appearing in the question portion of the same FAQ entity; $n(a, q_i)$ denotes the total number of FAQ entities, each of which contains the term a in its answer portion and the term q_i in its question portion at the same time; $n(q_i)$ denotes the total number of FAQ entities where the term q appears in the question portion.

Given a group of terms contained in the question portion, equation (2) shows how to estimate the chance of a term appearing in an answer portion and a group of terms contained in the question portion of the same FAQ entity, by separately computing the co-occurrence possibility of the term a in the answer portion caused by each term of the group of terms, $q_1, q_2, ..., q_n$, in the question portion in the equation (1).

Similarly, given a group of terms, $a_1, a_2, ..., a_m$, in the answer portion of any FAQ entity, we can compute the chance of a term, q, appearing in the question portion of the same FAQ entity in the same way, say,

$$p(q \mid a_j) = \frac{n(a_j, q)}{n(a_j)} \tag{3}$$

$$p_{q,\{a_1,a_2,\ldots,a_m\}} = 1 - \prod_{j=1}^{m} (1 - p(q \mid a_j)) \tag{4}$$

Where: $p(q|a_j)$ denotes the co-occurrence possibility of the term q appearing in a question portion caused by the term a_j appearing in an answer portion of the same FAQ entity; $n(a_j)$ denotes the total number of FAQ entities which contain the term a_j appears in their answer portions; The $p_{q,\{a_1,a_2,\ldots,a_m\}}$ can be interpreted as the chance of a term q appearing in the question portion of an FAQ entity caused by any of terms, a_1, a_2,..., a_m, contained in the answer portion of the same FAQ entity.

2.2 Procedures for Expanding an FAQ Entity

Following the basic strategy of the term expansion technique, the question portion and the answer portion of any FAQ entity in an FAQ collection can be expanded separately with particular terms with higher chances decided by the co-occurrence possibilities of the terms in a portion caused by any word contained another portion as follows.

(1) For each term appearing in the FAQ collection, count the number of FAQ entities that contain that term in their question portions and answer portions separately.

(2) For each two-term pair appearing in the FAQ collection, count the number of FAQ entities which contain the terms of the two-term pairs in the corresponding portions.

(3) Based on the equations (1) and (3), set up a list for storing the co-occurrence possibilities

of all two-term pairs appearing in the FAQ collection.

(4) Treating the corresponding question portion of each FAQ entity as a group of terms, construct an expanding term list for the answer portion of the same FAQ entity by using equation (2) to estimate the co-occurrence possibility of a term caused by a group of terms consisting of each term in the answer portion of each FAQ entity in the FAQ collection and the group of terms in the question portion of the corresponding FAQ entity.

(5) Similarly, treat the corresponding answer portion of each FAQ entity as a sentence, and construct an expanding term list for the question portion of the same FAQ entity by using equation (4) to estimate the co-occurrence possibility of a term in the question portion of each FAQ entity in the FAQ collection caused by a group of terms contained in the answer portion of the corresponding FAQ entity.

(6) Choose the top n terms which have greater chances in the term list as the extra terms for expanding a corresponding portion.

By doing so, each FAQ entity in the FAQ collection can be expanded with additional terms to provide extra lexical information for bridging the gap between two portions of any FAQ entity. Furthermore, by treating the user's query as an FAQ entity without the answer portion, a dummy answer for that user query can be constructed. Hence, a user query can be represented with much richer lexical information and can improve the possibility of correctly matched FAQ entity retrieval.

Using the procedure stated above, the complexity of the term expansion technique to bridge the lexical gap of two portions of each FAQ entity can be estimated in three parts. The complexity for counting the raw occurrence of the unique terms in FAQ entities is $O(M+N)$ where M is the total number of words appearing in the question portions of the collection and N is the total number of words appearing in the collection. The complexity for computing the co-occurrence possibility of each two-term pair appearing in the collection is $O(MN)$. The complexity for expanding the question and answer portions of an FAQ entity can be estimated as $O(2nm)$ in the worst case, where m and n are the number of unique terms contained in all question and answer portions of the FAQ collection, respectively. All in all, by assuming that $M >> m$ and $N >> n$ (unique terms can appear in an FAQ entity and a collection much more than one time), the complexity of expanding FAQ entities in an FAQ collection is $O(MN)$. Likewise, when treating the user query as an FAQ entity, the complexity for expanding a user query takes $O(nm)$.

By following the extra lexical information appended into the original FAQ entity, an FAQ entity could be represented as a term vector which is much richer with respect to the characteristics of the entity.

3 The DLSI Approach for the FAQ Retrieval Task

Following the basic strategy and procedures of the standard DLSI approach proposed by Chen [CTN01], procedures of the standard DLSI approach with necessary modifications is applied for solving an FAQ retrieval task rather than introducing the theory of the standard

DLSI approach itself.

3.1 Introducing the Approach for Solving the FAQ Retrieval Task in General

Following the basic theory of DLSI approach stated in Section 3.4 of Chapter II, for performing a FAQ retrieval task in general, the expanded question portion and corresponding answer portion of each FAQ entity is represented separately as two term vectors for independently describing the content of an FAQ entity. By establishing two term vectors for each FAQ entity in the FAQ collection, the two term-document vector spaces, the interior and exterior differential spaces, can be constructed for semantic analysis. As a characteristic of the DLSI approaches, the SVD method is employed for creating two lower order dimensional spaces, *k*-interior differential space and *k*-exterior differential space, in order to avoid problems regarding natural language in semantic analysis. Then, by treating a user's query as an FAQ entity without an answer portion, a likelihood function used in standard DLSI approach is applied for assessing the semantic similarity of the user's query and each single FAQ entity in the collection based on term-term and term-document relationships described separately in *k*-interior differential space and *k*-exterior differential space. Finally, following the standard DLSI approach, the posterior probability of an FAQ entity in the FAQ collection semantically relating to the given user's query is computed.

3.2 Procedures

3.2.1 Constructing the Term Vector

As the first step of setting up the DLSI approach for the FAQ retrieval task, the vectors of each FAQ entity should be established. For representing the content of an FAQ entity correctly, the terms in the FAQ collection are defined by the, stop-word list and word stemming techniques. By doing so, these two independent representations in the vector form for showing the contents of the expanded question and answer portions of each FAQ could be constructed by employing the log-entropy method for describing the importance of terms in an FAQ entity and in the FAQ collection based on their occurrences in different portions of an FAQ entity and in the FAQ collection. The weight of each term in an expanded question portion and the weight of each term in an expanded answer portion are calculated by two separate equations:

$$w_{q_i} = f(q_i) \cdot g_i , \tag{5}$$

$$w_{a_j} = f(a_j) \cdot g_j . \tag{6}$$

Here, the w_{q_i} denotes the weight of the i-th term, q_i, in the expanded question portion of an FAQ entity. The w_{a_j} stands for the weight of j-th term, a_j, in the expanded answer portion of an FAQ entity. The $f(q_i)$ and $f(a_j)$ denote the local weights of the term q_i appearing in the expanded question portion and the a_j in the expanded answer portion, respectively. The g_i and g_j are global weights of the terms q_i and a_j in the whole FAQ collection, respectively.

As applied in the standard DLSI approach, local weights of terms can be evaluated by logarithms of occurrence counts. Since lexical information in the corresponding portions of expanded FAQ entities has been changed by appending additional terms into FAQ entities, the local weights of terms in the expanded question portion and answer portion are calculated as follows:

$$f(q_i) = \log(1 + O_{q_i} + O_{Q_i}), \tag{7}$$

$$f(a_j) = \log(1 + O_{a_j} + O_{A_j}). \tag{8}$$

Where: the O_{q_i} and O_{a_i} are the raw occurrences of the terms q_i and a_j in the original question portion and corresponding answer portion of an FAQ entity separately; the O_{Q_i} and O_{A_j} are the occurrence times of the terms q_i and a_j in the appended terms of the expanded question and expanded answer portions separately as estimated by .

$$O_{Q_i} = \log\left(1 + P(q_i \mid a_1, \ldots, a_n) \times \frac{T_{q_i}}{d_{t_Q}}\right), \tag{9}$$

$$O_{Aj} = \log\left(1 + P(a_j \mid q_1, \ldots, q_m) \times \frac{T_{a_j}}{d_{t_A}}\right). \tag{10}$$

Here the $P(q_i \mid a_1, .., a_n)$ is the co-occurrence possibility of the term q_i appearing in the expanded question portion caused by any terms, a_1, a_2,…, a_n, in the original answer portion of an FAQ entity, $P(a_j \mid q_1, .., q_m)$ is the co-occurrence possibility of the term a_j appearing in the expanded answer portion caused by any terms q_1, q_2,…, q_m, in the original question portion, T_{q_i} is the total number of occurrences of the term q_i in the question portions of all original FAQ entities in the collection, and, d_{t_Q} is the number of FAQ entities which contain

the term q_i in their original question portions. The ratio of T_{q_i} and d_{tq} can be understood as the relative number of occurrences of the term q_i, as an additional term appended into the question portion of an FAQ entity. Similarly, T_{a_j} is the total number of occurrences of the term a_j in the answer portions of all original FAQ entities in the collection, and d_{tA} is the number of FAQ entities which contain the term a_j in their original answer portions. Thus, the ratio of T_{a_j} and d_{tA} can be understood as the relative number of occurrences of the term a_j in the answer portions, if it appears as an additional term. Hence, the O_{Q_i} could be interpreted as the weight of possible occurrences of the term q_i smoothed by logarithm if it appears in a question portion as an additional term. Meanwhile, O_{A_j} is the weight of possible occurrence times of the term a_j smoothed by logarithm.

Furthermore, the global weights of terms in the question portion and answer portion of each FAQ entity in the collection can be decided by the entropy weighting method as

$$g_i = 1 - \frac{1}{\log N} \sum_{j=1}^{N} p_{ij} \log(p_{ij}). \tag{11}$$

Where N is the number of FAQ entities in the collection, and, p_{ij} is the possible of term t_i appearing in the FAQ entity j and estimated by

$$p_{ij} = \frac{O_{ij}}{T_i}. \tag{12}$$

Where O_{ij} is the number of occurrences of the term t_i in the FAQ entity j, and T_j is the number of occurrences of the term t_i in the FAQ collection. Here, define $p_{ij} \log(p_{ij}) = 0$, if $p_{ij} = 0$.

Finally, for comparing all expanded question and answer vectors of all FAQ entities in a

unified standard, each vector V_i ($e_1, e_2, ..., e_k$) is normalized as $V_i'(n_1, n_2, ..., n_k)$ with

$$n_i = e_i \Big/ \sqrt{\sum_{j=1}^{k} e_j^2} \ . \tag{13}$$

Hence, the term vectors for correctly describing contents of expanded FAQ entities could be established by applying the equations, presented above, in procedures as follows.

(1) For calculating the local weight and global weight of terms, the number of occurrences of terms in two parts of each single FAQ and all FAQs in the collection are counted one by one.

(2) Compute the global weight of each term appearing in the collection by equations, (11) and (12) respectively based step (1).

(3) For each FAQ entity in the collection, calculate the local weights of the terms appearing in the question portion by equation (7) and in the answer portion by using equation (8), and then the weights of the expanded term vectors for the expanded question portion based on equation (5) and the expanded answer portion based on equation (6).

(4) Finally, for comparing all vectors in a standard length, normalize the expanded term vectors through equation (13).

By procedures for constructing the term vectors for FAQ entities given above for estimating the complexity of constructing term vectors for an FAQ entity, assume that an FAQ collection contains *t* unique terms and *d* FAQ entities where the answer portions of the collection have *M* terms in total with *m* unique terms. Question portions of the collection have *N* terms in

total and n unique terms. Since unique terms can appear in question and answer portions more than one time, the $M >> m$ and $N >> n$, the complexity for counting the raw occurrence of terms in the collection is $O(N+M)$. For computing global weights, the complexity is $O(td)$. For calculating local weights, the complexity is $O(2t)$. Finally, for normalizing each term-document vector, the complexity is $O(td)$. Thus, the complexity for constructing the term-document vectors of each FAQ entity in the FAQ collection is estimated as $O(N+M+td)$.

At this point, the question and answer portions of each expanded FAQ entity in an FAQ collection can be clearly represented as two weighted and normalized vectors. As a result, semantic analysis spaces, interior differential space and exterior differential space, can be created.

3.2.2 Setting up the Semantic Analysis Space

As distinguished from the standard LSI method, the DLSI method evaluates similarity between a given user query and each FAQ entity based on two term-document vector spaces, interior differential space and exterior differential space, which reflect the particular characteristics of an FAQ entity in the lexical differences between two different portions of that entity (internal difference) and the lexical differences of any two FAQ entities (external difference) based on the totality of terms of the whole FAQ collection.

By representing the contents of each FAQ in two independent term-document vectors

constructed by the expanded answer portion and the expanded question portion, the interior differential space can be generated by the interior differential vectors, $I^{(in)}_{i=1...d}$, of the d FAQ entities in the collection. An interior differential vector is defined by the difference of the term-document vector of expanded question portion and the term-document vector of the corresponding expanded answer portion according to the equation:

$$I_i^{(in)} = V_{i,Q} - V_{i,A} . \tag{14}$$

where $V_{i,Q}$ and $V_{i,A}$ are the term-document vectors of the expanded question portion and expanded answer portion of the i-th FAQ entity respectively. As a result, the basis for representing the interior differential space, D_I, is constructed by the interior differential vectors of each FAQ entity in the FAQ collection as $D_I(I_1^{(in)}, I_2^{(in)}, ..., I_n^{(in)})$.

In the same way, the exterior differential vector, $I_{ij}^{(ex)}$, of any two FAQ entities, is similarly established by using the differences of the vector of the expanded question portion of the i-th FAQ entity and the vector of the expanded answer portion of the j-th FAQ entity as,

$$I_{i,j}^{(ex)} = V_{i,Q} - V_{j,A} . \tag{15}$$

where $V_{i,Q}$ is the term-document vector of the expanded question portion of i-th FAQ entity, and $V_{j,A}$ is the term-document vector of expanded answer portion of the j-th FAQ entity. Likewise, the basis for representing the exterior differential space, D_E, is constructed by all exterior differential vectors as $D_E(I_{1,2}^{(ex)}, I_{2,3}^{(ex)}, ..., I_{n-1,n}^{(ex)})$.

As in the LSI method, the DLSI method also applies the SVD method as shown in equation 16 to obtain two lower dimensional spaces, k-interior differential space and k-exterior

differential space, of which the dimensions are truly independent conceptual meanings, by exploring orthogonal dimensions in the original high dimensional spaces.

$$D = USV^T \approx U_k S_k V_k^T = D_k . \tag{16}$$

Where: D is the original interior differential space or the exterior differential space constructed by n differential vectors each of which has m dimensions. The r is the rank of D and $r \le n$. U and V are the eigenvectors of DD^T and $D^T D$ respectively. $S = diag(\delta_1, \delta_2, ..., \delta_m)$, where δ_i are non-negative square roots of eigen values of DD^T, $\delta_i > 0$ for $i \le r$ and $\delta_i = 0$ for $i > r$; Similarly, D_k is a lower dimensional space in $m \times k$ and approximately equal to D. U_k and V_k are the eigenvectors of $D_k D_k^T$ and $D_k^T D_k$ respectively, $S_k = diag(\delta_1', \delta_2', ..., \delta_k')$ where δ_i' are nonnegative square roots of eigen values of $D_k D_k^T$, $\delta_i' > 0$. The k is various from different collections used and usually decided by corresponding experiments. As the experiences show in the related reference [CTN01], "Generally $k \ge 100$ will be selected for $1000 \le n \le 3000$, and the corresponding k is normally smaller for the interior differential term-document matrix than that for the exterior differential term-document matrix".

The lower dimensional bases of interior differential space and exterior differential space can be decided by the following procedures.

(1) By constructing term-document vectors of the expanded question portion and expanded answer portion of each FAQ entity in the collection as described above, the interior differential vector of each FAQ entity is defined by equation (14). As a result, the interior

differential space is constructed.

(2) Similarly, an exterior differential vector of the i-th FAQ entity and (i+1)-th FAQ entity is defined by equation (15). By constructing the exterior differential vectors for covering all FAQ entities in the FAQ collection, the exterior differential space is also established.

(3) For analyzing term-term relationships and term-document relationships, the two lower dimensional spaces, k-interior differential space and k-exterior differential space, are found by using the SVD method to decompose the basis with the equation (16).

Complexity for constructing the k-interior differential space and k-exterior differential space without considering the estimation on applying SVD method is $O(2td)$ where d is the number of FAQ entities in the collection, and t is the number of the terms in the collection.

3.2.3 Evaluating Semantic Similarity

For discovering FAQ entities semantically related to a user's query, treating the user's query as an FAQ entity with the question portion only and following exactly the same procedure for expanding the answer portion of an FAQ entity as stated in Section 3.1, any expanded user's query containing the original user's query and a dummy answer can be represented as a vector V_q. For representing the differences between a user's query and the *i-th* FAQ entity in the collection, a differential term-document vector, x_i, could be formulized as

$$x_i = V_q - V_i \tag{17}$$

where V_q is the term-document vector of the expanded user's query. V_i is the

term-document vector of the *i-th* FAQ entity with the expanded question and answer portions.

By constructing the differential term vectors of an expanded user query and each FAQ entity in the collection, as the equation stated in the standard DLSI method, the likelihoods of the user query and each FAQ entity described in the *k*-interior differential space $P(x_i|D_I)$ and *k*-exterior differential space $P(x_i|D_E)$ are, respectively

$$\hat{P}(x_i \mid D_I) = \frac{n^{1/2} \exp\left(-\frac{n}{2}\sum_{i=1}^{k}\frac{y_i^2}{\delta_i^2}\right)\cdot \exp\left(-\frac{n\varepsilon^2(x_i)}{2\rho}\right)}{(2\pi)^{n/2}\prod_{i=1}^{k}\delta_i \cdot \rho^{(r_{D_I}-k)/2}}, \tag{18}$$

$$\hat{P}(x_i \mid D_E) = \frac{n^{1/2} \exp\left(-\frac{n}{2}\sum_{i=1}^{k}\frac{y_i^2}{\delta_i^2}\right)\cdot \exp\left(-\frac{n\varepsilon^2(x_i)}{2\rho}\right)}{(2\pi)^{n/2}\prod_{i=1}^{k}\delta_i \cdot \rho^{(r_{D_E}-k)/2}} \tag{19}$$

where $y = (y_1, y_2, ..., y_k)^T = U_k^T x_i$, $\varepsilon^2(x_i) = \|x_i\|^2 - \sum_{i=1}^{k} y_i^2$, $\rho = \frac{1}{r-k}\sum_{i=k+1}^{r}\delta_i^2$, r_{D_I} and r_{D_E} are the ranks of matrix D_I and D_E respectively, x_i is a differential term vector of an expanded user's query and an expanded FAQ entity defined by equation (16) above.

Hence, the similarity of a user query and each FAQ entity in the FAQ collection is measured by the posteriori probability

$$P(D_I \mid x_i) = \frac{P(x_i \mid D_I)P(D_I)}{P(x_i \mid D_I)P(D_I) + P(x_i \mid D_E)P(D_E)} \tag{20}$$

where the prior probability $P(D_I)$ is set to be the average number of FAQs returned to the user divided by the number of FAQ entities in the FAQ collection and $P(D_E)$ is set to be $1 - P(D_I)$.

In general, the semantic similarity of a user's query and each FAQ entity in the collection is

based on two lower dimensional spaces discovered through the following procedures.

(1) Treated as an FAQ entity without the answer portion, the user query is expanded by using term expansion techniques stated in section 2.2 to create a dummy answer.

(2) Build the vector of a user query V_q by following the procedure of constructing term vectors as stated in section 3.3.1.

(3) Construct the differential vectors, $x_{i=1,\ldots,d}$, by using the user query vector V_q and the vectors of each FAQ entity in the FAQ collection, $V_{i=1,\ldots,d}$, based on equation (17).

(4) By using each differential vector x_i the likelihoods of the user query and the i-th FAQ entity described in the k-interior differential space and in the k-exterior differential space are calculated using equations (18) and (19) respectively.

(5) As a result, semantic similarities between the user query and all FAQ entities in the collection are evaluated by equation (20).

(6) Choose n FAQ entities with the highest semantic similarities as candidates and return to a user.

Following the procedures for expanding a user's query and constructing a term-document vector with t dimensions for the query, the complexity is $O(nt)$ in the worst cases. Consequently, the complexity for constructing differential vectors by using the term-document vector of a user query and each term-document vector of d FAQ entities is $O(td)$. For calculating the similarities of a user query and each one of d FAQ entities based

on the two lower dimensional spaces, the complexity is $O(dt^2)$. In summary, by using the quick sort algorithm, the complexity of which is $O(d^2)$ in the worst case, to sort candidates based on similarities of the user query and all d FAQ entities in the collection, the complexity for retrieving candidates semantically related to a user query is estimated as $O((t+d)^2)$.

4 Template Matching and Merging

The DLSI method shows an extraordinary ability [CTN03, CTN01] to evaluate similarity of any two documents based on addressing the latent semantic relationships of terms appearing in the whole document collection. However, the real desires of users are only partly reflected by the ranks of corresponding candidates, especially in FAQ retrieval tasks, due to vague meaning represented by limited lexical information contained in both users' queries and FAQ entities. For example, given a user query, "*What are the main objects for indexing and searching?*" three FAQ entities, "*What is indexing?*", "*What is searching?*", and "*What are terms?*", could be retrieved as candidates in semantically related levels computed by the DLSI approach. By applying human intelligence to understand the query, the FAQ entity, "*What are terms?*", can easily be identified as the best match for the user query because terms are the main objects for indexing and searching in IR topics. Consequently, a method for pinpointing what users really want in a candidate list is still required.

Another observation on FAQ retrieval tasks is that the contents of corresponding FAQ entities usually contain few sentences syntactically similar to a user query, such as the sentence, "*...terms are the basic units for indexing and searching*", contained in the FAQ entity, "*what*

are terms?". Hence, evaluating the syntactic similarity of a user query and an FAQ entity, such as similarity of word order of the same terms appearing in both query and FAQ entity, could assist to further narrow down what the user wants from few semantically related FAQ entities since the syntactic information provides a description of relevance of a user query and an FAQ entity from another viewpoint.

Inspired by the template automation system originally developed for an ILTS (intelligent language tutoring system) [CT03], which shows a powerful ability to represent differences between simple grammatically valid sentences with synonymous expressions, the template structure with its matching algorithm is useful for addressing similarities between any user query and each sentence of an FAQ entity by discovering terms that match exactly in the same word sequence.

4.1 The Structure of a Template

For illustrating word order of terms in any sentence of an FAQ entity, a template is usually defined as a directed acyclic graph (DAG) which contains a few sentences related to the same topic by sharing the same words of different sentences in particular rules. In a template, each word is called a vertex and is connected to other vertices by edges. Hence, specifications of two basic objects, vertex and edge, of a template are given as follows.

4.1.1 The Specification of Vertex

Each vertex of a template represents a stop-word or term appearing in a sentence of an FAQ

entity. Each vertex is described by three fields, *uid*, *content*, and *type*:

- *uid* is a unique identification of a term or stop-word in a template and contains two subfields, the id of a sentence (subpath) contained in a template and the sequence index of a term or stop-word in a sentence, such as "*V2:1*" representing the first term/stop-word of the second sentence in a template;
- *content* is a stop-word or a term in stemmed form represented by a vertex;
- *type* is the type of word: 0 represents stop-word and 1 for term.

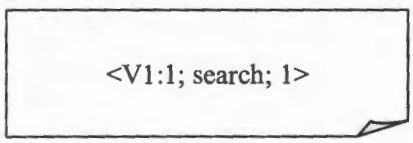


Figure 4: A Vertex

Consequently, the vertex entity, as shown in Figure 4, means that the first vertex of the second sentence contained in the template represents as a term the stemmed form of "search". For representing a template expediently, two special vertices, the start vertex and the terminal vertex, are defined as "Vs; -; 0" and "Ve; ?; 0" respectively and shared by all sentences in a template. Hence, any sentence could be represented by a group of vertices, or a vertex list, which contains all the terms and stop-words appearing in the original sentences. For example, the sentence, "*what is searching*", as the only sentence contained in a template, could be represented in a vertex list as in Figure 5.

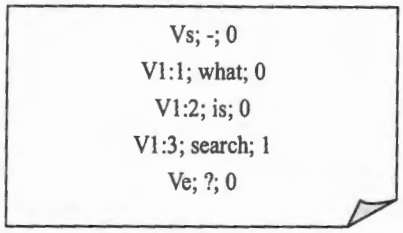


Figure 5: A Sentence in a Vertex List

4.1.2 The Specification of an Edge

Each edge of a template connects two vertices in word order of the two words shown in the original sentence, and could be represented as a pair of *uid*s of two vertices, "<V1:1, V1:2>". Hence, each sentence contained in a template, also called a subpath, could be represented as a group of edges, an edge list, arranged in the word order of terms or stop-words appearing in the original sentence. By following the vertices defined in Figure 5, the sentence, "*what is searching?*", could be represented in an edge list in Figure 6.

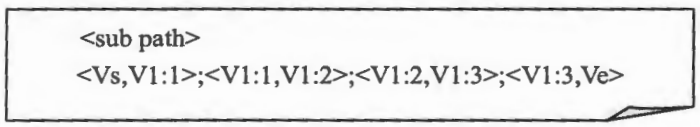


Figure 6: One Sub Path in the Edge List of a Template

4.1.3 The Specification of Template

Described by a vertex list and a corresponding edge list, a template represents a group of subpaths which share the same starting/terminal vertex and the vertices decided by particular rules. A template contains at least one subpath which contains two vertices, a starting vertex and a terminal vertex. For example, the sentence in Figure 6 can be defined as a template

which contains one subpath (sentence) only and is shown visually in DAG form in Figure 7.

Figure 7: One Sub Path in a Template Shown in DAG Form

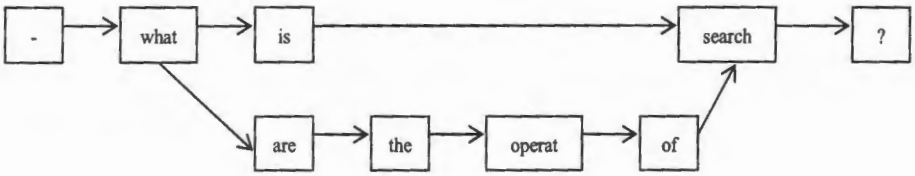


Figure 8: A Template with Multiple Sub Paths

A template usually contains two or more subpaths related to the same topic by sharing the same terms or stop-words in a few rules, such as particular word order. For example, in Figure 8, a template containing two subpaths of the sentences, "*what is searching*?" and "*what are the operations of searching*?", could be described by the vertex list and edge list shown in Figure 9.

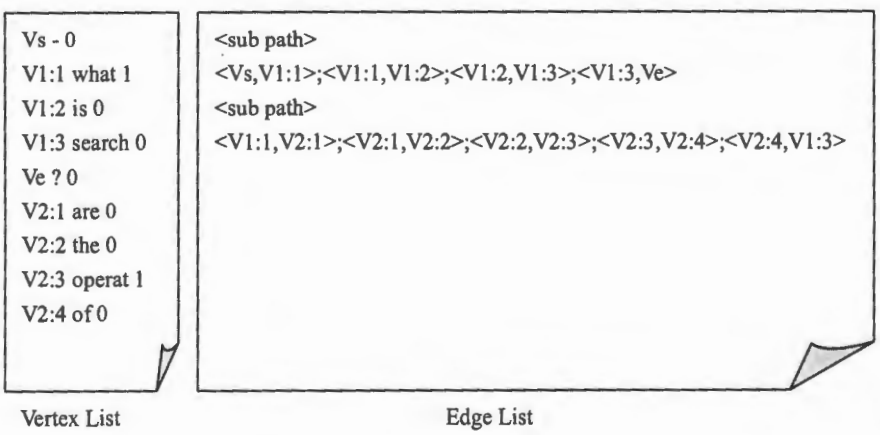


Figure 9: The Definition of a Template

4.1.4 Procedures of Representing a FAQ entity as Template

Following the specifications of a template with its two essential objects, vertex and edge, any one of the FAQ entities in the collection can be represented as two independent templates: the template of a question portion and the template of an answer portion in the following procedures.

(1) To define each vertex and edge of the question portion of an FAQ entity, read the words $\{w_1, w_2, ..., w_i, ..., w_n\}$ in the question portion one by one:

a) construct two vertices, $<Vs;$ -; $0>$ and $<Ve;$?; $0>$;

b) if w_i is a stop-word, then construct a vertex entity as $<V1:i, w_i, 0>$;

c) if w_i is a term, then stem w_i into w_i' and create a vertex entity as $<V1:i, w_i', 1>$;

d) if w_i is the first word appearing in the question sentence, then create an edge $<Vs, V1:1>$;

e) if w_i is not the first word, then construct an edge entity $<V1:i-1, V1:i>$,

i. if w_i is the last word appearing in the question sentence, then construct a edge entity $<V1:i, Ve>$.

(2) For defining each vertex and edge of the answer portion of an FAQ entity, read the words, w_{ij}, in the j-th sentence, of answer portion of a FAQ entity one by one

a) construct two vertices, $<Vs;$ -; $0>$ and $<Ve;$?; $0>$;

b) if w_{ij} is a stop-word, then construct a vertex entity as $<Vj:i,w_{ij},0>$;

c) if w_{ij} is a term, then stem w_{ij} as w_{ij}' and create a vertex as $<Vj:i,w_{ij}',1>$;

d) if w_{ij} is the first word appearing in the question sentence, then construct an edge entity $<Vs,Vj:1>$;

e) if w_{ij} is not the first word, then create an edge $<Vj:i-1,Vj:i>$,

i. if w_{ij} is the last word appearing in the question sentence, then construct an edge entity $<Vj:i,Ve>$.

Assuming an FAQ entity contains *m* and *n* words respectively, the complexity for translating an FAQ entity into two templates is $O(m+n)$.

An FAQ entity can be translated into two independent templates for representing its corresponding portions. The syntactic similarity of templates can be studied by a template matching algorithm, such as the HCS (heaviest common sequence) algorithm of ILTS [CT03] which aims to discover a word order in subpaths of any two templates by calculating preset weights of exactly matched words in the same sequence [CT03].

4.2 Template Matching Algorithm

For the FAQ retrieval task, by treating a user query as an FAQ entity with the question portion only, the syntactic similarity between a user query and an FAQ entity can be addressed. This can be accomplished by computing the similarity in word order of terms

and stop-words between the template of a user query and templates of two portions of an FAQ entity based on a template matching algorithm. Here, the template matching algorithm as a simple version of HCS algorithm considers matching a template with one subpath and a template with multiple subpaths rather than considering matching two templates with multiple subpaths in an HCS algorithm.

4.2.1 Two Factors for Computing the Syntactic Similarity

For finding the similarity between a user query and any portion of an FAQ entity by comparing two templates defined above, two factors, the *score* of longest common sequence and *distance* of terms appearing in the longest common words sequence, are introduced for illustrating the syntactic relationship between any two templates.

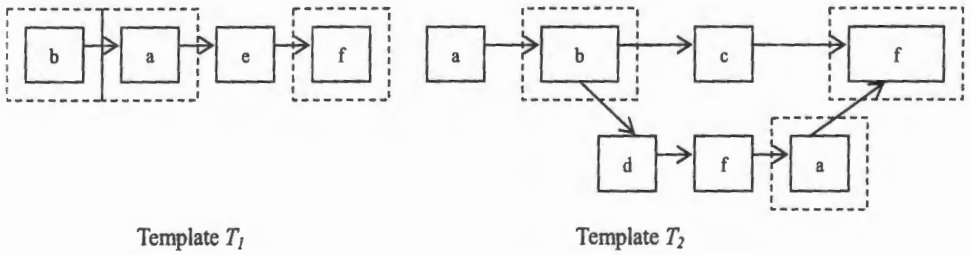


Figure 10: Two Templates Sharing the Common Terms "*b a f*"

The longest common sequence (LCS) of two templates is an exactly matched vertex sequence which has the maximum number of vertices, each of which represents a stop-word or a term and is in a same subpath (sentence), and is found by comparing the vertex sequence of others contained in the two templates. Hence, the *score* of an LCS is defined as the number of words contained in the LCS of the two templates. For example, in Figure 10, the *score* of

the templates T_1 and T_2 is defined as 3 since the longest common sequence "*b a f*" contains three words in the same occurrence order. The factor of the *score* gives a direct quantitative description of how many of the same words of the two templates appear in particular word order. The higher the *score* two templates have, the stronger the relevance.

The factor *distance* between the templates, T_1 and T_2, is defined as the minimum number of terms for changing a subsequence of T_1 and a subsequence of T_2 into the terms sequence of their LCS. The subsequence as a part of a subpath in a template is defined as the sequence of a subpath starting from the vertex which is the first term of the LCS and terminating at the vertex of the template which is the last term of the LCS. For example, in Figure 10, by regarding all words in the two templates as stop-words except *b*, *a* and *f*, and the longest common sequence as "*b a f*" of template T_1 and T_2, the subsequence of template T_1 is "*b a e f*". Likewise, the subsequence of template T_2 is "*b d f a f*". Hence, the *distance* is the number of modification for changing the subsequence of T_1 and T_2 into "*b a f*"", the term sequence of LCS. Rules for computing the *distance* between any two templates are described as follows.

(1) As the initial value of *distance*, the *distance* of any two templates is 0.

(2) If the deleted word is a term, the *distance* is increased by 2.

(3) If the deleted word is a stop-word, the *distance* is increased by 1.

(4) If no deleting is required, the *distance* is increased by 0.

(5) If there is only one term in the longest common sequence, the *distance* is defined as -1.

As in Figure 10, if all words in the two templates are stop-words except *b*, *a* and *f*, the distance of template T_1 and T_2 is 4 due to deleting one stop-word *e* in the template T_1 and deleting one stop-word *d* and one term *f* in the template T_2. Since terms carry more semantic information than stop-words, the factor *distance* can emphasize the importance of connections among terms in an LCS rather than consider stop-words equally as important as the terms. For example, given the three sentences, *S1*, *S2* and *S3*, and the corresponding templates, as shown in Figure 11, the problem for distinguishing these three semantically related sentences can be solved.

S1: How can I operate a searching task?

S2: What are the operations behind searches?

S3: Why should we operate the re-indexing job after amount of searching?

By sharing the longest common sequence "*oper search*" where each term is stemmed by the standard Porter Stemmer as described in Section 3.2 of Chapter II, the value of the *score* of T_3 and T_4 and the value of the *score* of T_3 and T_5 are the same: 2. Consequently, one can suffer from making a distinction on their syntactic similarity based on factor *score* only. The factor *distance* is introduced as the differences can be clearly pointed out due to difference between the *distance* of T_3 and T_4 being 2 and the *distance* between T_3 and T_5 being 10. Hence, by considering these two factors relatively, syntactic similarity of any two templates can be discovered based on how many exactly matching words appear in the same sequence as well as on the differences of terms in the LCS represented in the two different templates.

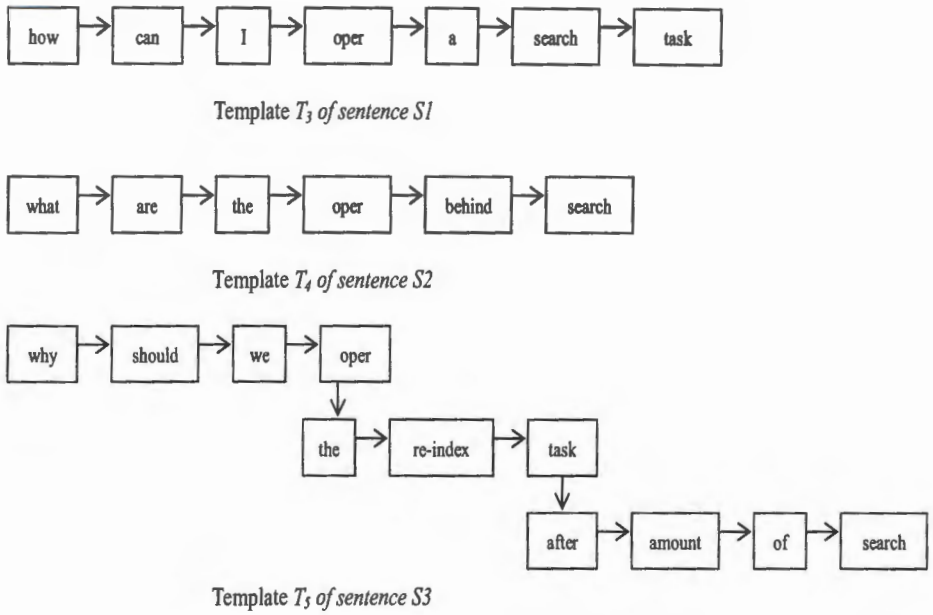


Figure 11: Three Templates with the same score and different distance

4.2.2 Calculating the Syntactic Similarity Based on Dynamic Programming

As an algorithm design technique employed for discovering LCS (Longest Common Subsequence), the problem of evaluating similarity of any two information sequences representing the knowledge of particular fields, such as DNA sequences, can be solved by finding the maximum number of information units which exactly match in the same occurrence order in any two information sequences. The dynamic programming technique uses an additional data structure, such as a table or multidimensional array, to represent the optimal value for a particular sub-problem in each entry of the table. By applying a two dimensional array $a[m, n]$ to represent the *score* of LCS of two strings, $s_1(t_1,t_2,...,t_m)$ and $s_2(t_1,t_2,...,t_n)$, the algorithm for solving the LCS problem based on dynamic programming is

usually designed for calculating the current score of the LCS of the substring $t_1,t_2,...,t_i$ of s_1 and the substring $t_1,t_2,...,t_j$ of s_2 in a cell $a[i,j]$ of the table, a two-dimensional array, by comparing the values of the length of the longest common sequence held by the diagonal cell $a[i-1,j-1]$ and the neighbor cells, $a[i-1,j]$ and $a[i,j-1]$.

Since all vertices belonging to multiple subpaths of a template could be arranged in linear order by applying the topological algorithm[2], a similar method for discovering the LCS of two templates by setting a two dimensional array, $TM[m,n]$, as the data structure is applied for computing the longest common sequence of two templates based on dynamic programming technique. Here, for convenience of algorithm description, each cell $TM[i,j]$ of $TM[m,n]$ contains three fields, *score*, *distance*, and *common sequence pairs* (*csp*), for describing syntactic similarity of two sub-templates of a user query and sub-templates of an FAQ entity. The two sub-templates described in cell $TM[i,j]$ of the array are defined separately as template, T_i^{uq}, which contains all subpaths of the query template constructed by vertices from the starting vertex of the query template to its *i*-th vertex, and a template, T_j^{faq}, which contains all subpaths of an FAQ entity template constructed by vertices from the starting vertex of the FAQ entity template to its *j*-th vertex. Additionally, vertex v_i, which directly connects to v in a template, is called a predecessor vertex of the vertex v_i. Furthermore, by regarding the current cell $TM[i,j]$ as the representation of the syntactic

[2] A topological sort of a DAG G = (V, E) is a linear ordering of v ∈ V such that if (u, v) ∈ E then u appears before v in this ordering.

similarity of the sub-template T_i^{uq} of the query template and the sub-template T_j^{faq} of the FAQ entity template, the element $TM[i,j']$, named the left predecessor cell of $TM[i,j]$, represents the syntactic similarity of the sub-template T_i^{uq} of the query template and the sub-template $T_{j'}^{faq}$ of the FAQ entity template which contains all subpaths of an FAQ entity template constructed from the vertices from its first vertex to the vertex $v_{j'}$, a predecessor vertex of the vertex v_j. Similarly, the cell $TM[i',j]$ is called the top predecessor cell of $TM[i,j]$, which represents the syntactic similarity of sub-template T_j^{faq} of the FAQ entity template and sub-template $T_{i'}^{uq}$ of the query template which contains all subpaths of a user query template constructed from its first vertex to the vertex $v_{i'}$, the predecessor vertex of the vertex v_i. Consequently, the cell $TM[i',j']$ is called the left-top predecessor cell of $TM[i,j]$. It also represents the syntactic similarities of sub-template $T_{j'}^{faq}$ of the FAQ entity and sub-template $T_{i'}^{uq}$ of the query template.

By defining related notations as above, syntactic similarity of any two sub-templates of a query template and an FAQ entity template described by the optimal value of $score_{i,j}$ and $distance_{i,j}$ held by the current cell $TM[i,j]$ could be calculated by the following procedure.

(1) If the words represented by vertices, v_i and v_j, are exactly same, add 1 to $score_{i',j'}$ where $score_{i',j'}$ is the maximal value of *score* held by all left-top predecessor cells of the current cell $TM[i,j]$. Then compare the values of $score_{i,j}$, $score_{i,j'}$, and $score_{i',j}$. The $score_{i,j'}$ is the maximal value of *score* held by one of the left predecessor cells of $TM[i,j]$. The $score_{i',j}$ is maximal value of *score* held by one of the top predecessor cells of

$TM[i, j]$.

a) If $score_{i,j'}$ or $score_{i',j}$ is greater than $score_{i,j}$, then the current cell inherits the values of all fields from the predecessor cell which holds the maximal value of $score_{i,j'}$ or $score_{i',j}$. If the maximal value of $score_{i,j'}$ or $score_{i',j}$ is held by more than one predecessor cell, then the current cell inherits the corresponding values of the other two fields from the left predecessor cell or the top predecessor cell which holds the minimal value of *distance* among the left predecessor cells and top predecessor cells which hold the maximal value of *score*. After that, modify the value of *distance* of the current cell, following the rules stated in section *4.2.1*.

b) If $score_{i,j}$ is greater than $score_{i,j'}$ or $score_{i',j}$, then the values of other fields of the current cell are inherited from the left-top predecessor cell which holds the value of $score_{i',j'}$. If the value of $score_{i',j'}$ is held by more than one cell, the values of other fields of the current cell are inherited from the cell which holds the minimal value of *distance* among the left-top predecessor cells which hold the value of $score_{i',j'}$. After that, modify the value of *distance* of the current cell following the rules stated in section *4.2.1*, and append the matching pair $mp_{i,j}(v_i, v_j)$ which contains the two corresponding vertices v_i and v_j in two templates, into the *csp* held by the current cell.

c) If $score_{i,j}$ equals $score_{i,j'}$ or $score_{i',j}$, then the current cell inherits the values from the predecessor cell which holds the minimal value of *distance* among the

predecessor cells which hold the value of $score_{i,j}$, $score_{i,j'}$, or $score_{i',j}$. If a left-top predecessor cell is picked up, then modify the value of *distance* of the current cell by following rules described in *section 4.2.1*, and append the matching pair $mp_{i,j}(v_i, v_j)$ into the *csp* held by the current cell.

(2) If the words represented by vertices, v_i and v_j, are not same, compare the *score* value held by the top predecessor cells and the *score* value held by the left predecessor cells. Then, the current cell inherits the values of all fields from the cell which holds the maximal value of the *score*. If the maximal value of the *score* is held by more than one left or top predecessor cell, then the current cell inherits the corresponding values of all fields from the predecessor cell which holds the minimal value of *distance* among the left predecessor cells and top predecessor cells which hold the maximal value of *score*. After that, modify the value of *distance* of the current cell, following rules stated in section *4.2.1*.

By calculating each cell of the two dimensional array, syntactic similarity between the template of a user query and template of an FAQ entity could be found in the bottom right cell $TM[m,n]$. By assuming the template of a user query and template of an FAQ entity contain m and n vertices separately, the complexity for calculating syntactic similarity between the template of a user query and template of an FAQ entity is $O(mn^2/2)$ in the worst case without considering the particular data structures and algorithms for storing and searching the corresponding descriptions of vertices and edges.

Furthermore, the candidates in a candidate list containing a few semantically related FAQ entities filtered by the DLSI approach could be reordered to reflect what the user really wanted based on syntactic similarity between the user query and each candidate by applying the template matching algorithm.

4.3 Template Merging Algorithm

Two characteristics of the template structure are that different representations on the same topic can be organized in one template by sharing the same start vertex and terminal vertex, and each vertex can be decided by particular rules, such as the stop-word or term which belongs to the longest common sequence of any two subpaths. In other words, the more representations a template contains, the more comprehensively lexical expressions on the same meaning can be covered. Thus, syntactic similarity between an upcoming user's query and an FAQ entity can be caught more precisely. Hence, any template can be expended with related user's queries as various representations of the same FAQ entity. In reference to *common sequence pairs* of any two templates generated in the template matching procedures, the template of a user query could be easily merged into the template of a corresponding FAQ entity. The procedure of merging template T_b into template T_a directly relates to modifications to specifications of the template T_a by adding the definitions of vertices and edges into the corresponding vertex list and edge list.

For example, template T_b, which has n_b vertices, is merged into a template T_a, which has n_a vertices, by giving the longest *common sequence pairs*, *csp*, which contains *n* matching

pairs, $mp(v_i, v_j)$, where v_i is the i-th vertex belonging to template T_a, and v_j is the j-th vertex belonging to template T_b. The merging procedures could be described as follows.

(1) If $csp = \phi$, append definitions of vertices and edges of T_b into corresponding lists of T_a with two additional directed edges, an edge linking the start vertex, v_s, of template T_a to the first vertex of template T_b and an edge linking the last vertex of template T_b to the terminal vertex, v_e, of template T_a.

(2) If the number of *mp* in *csp* equals the number of vertices of a subpath of template T_a and the number of vertices of a subpath of template T_b, no modification to the specifications of T_a is required for the definition

(3) Define the temporary start vertex, V_{ts}, and temporary terminal vertex, V_{te}, for merging each vertex, v_b, of the template T_b into the template T_a one by one.

a) If v_b is the first vertex of the template T_b and not in the first matching pair of *csp*, mp_1, then, set V_{ts}, as v_s and V_{te}, as v_b.

b) If v_b is the first vertex of the template T_b and in the $mp_1(v_a, v_b)$ of *csp*, then set V_{ts} as v_s, and V_{te}, as v_a.

c) If v_b is not the first vertex of the template and not in any $mp_i(v_a, v_b)$ of *csp*, then set V_{ts} as V_{te} and V_{te} as v_b.

d) If v_b is not the first vertex of the template T_b and in a $mp_i(v_a, v_b)$ of *csp*, then set V_{ts} as V_{te} and V_{te} as v_a.

e) Register a direct edge linking V_{ts} to V_{te} and the temporary terminal vertex V_{ts} into template T_a, if the definitions of the direct edge and temporary terminal vertex are not presented in the vertex list or edge list of template T_a.

(4) By appending the vertices of template T_a, register a direct edge linking V_{te} to the terminal vertex of template T_a if the definition of the direct edge does not exist in the vertex or edge list of template T_a.

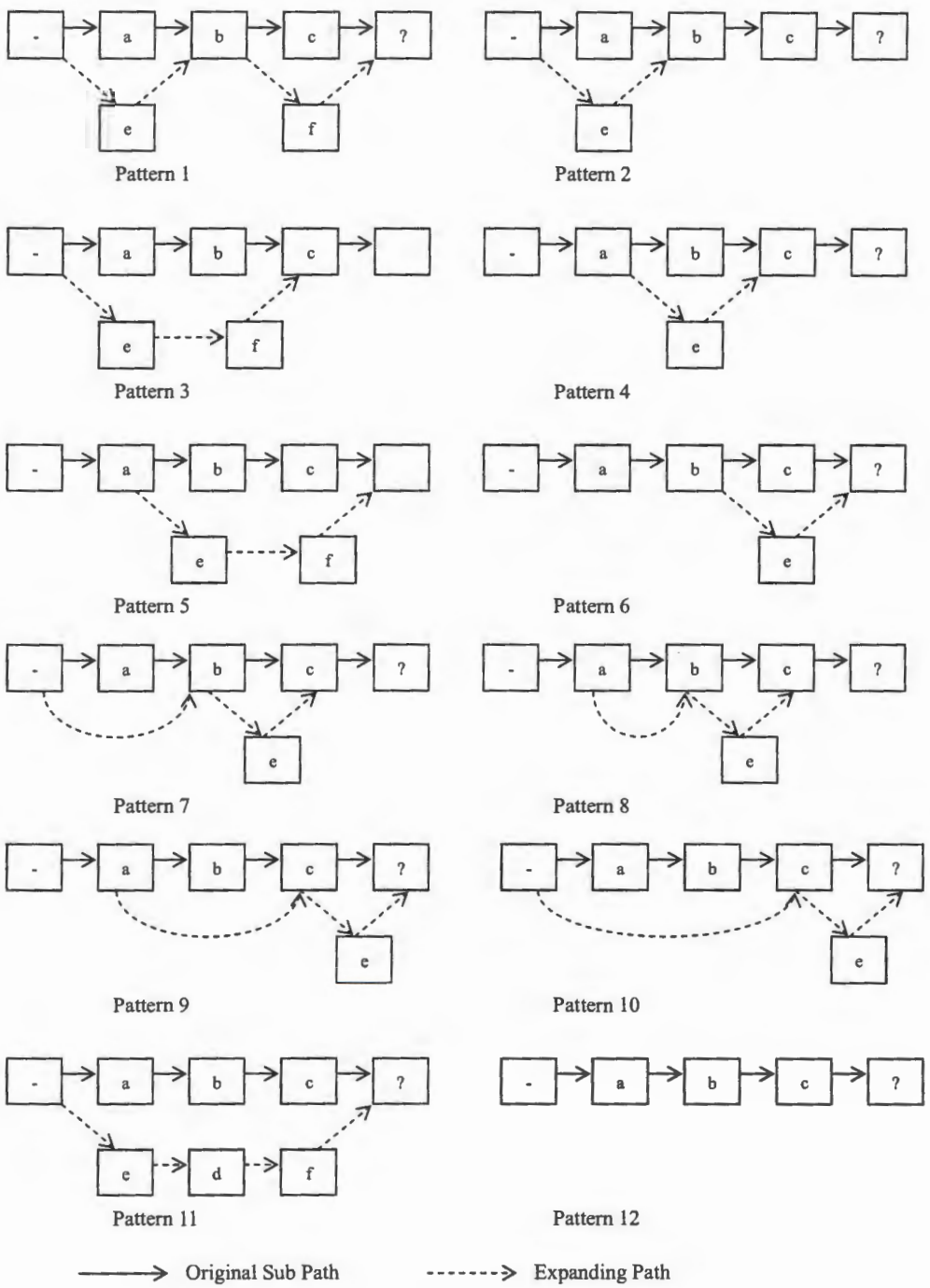


Figure 12: Merging Patterns

Following the template merging method stated above, possible merging patterns are covered in Figure 12 for appending a template of a user query into a template of an FAQ entity through registering necessary vertices and corresponding edges into the templates. As a result, the complexity of the *template* merging method could be estimated as $O(n_b)$ in the worst case without considering particular data structures and algorithms for storing and searching the corresponding descriptions of vertices and edges, where n_b is the number of vertices in template T_b.

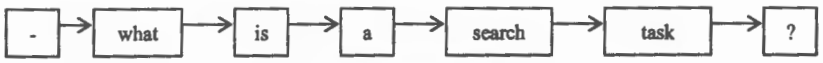
The procedure of appending users' queries into corresponding templates based on the decisions of users is also called Dynamic Template Expansion (DTE). By doing DTE day by day, the ability of a template to catch syntactic similarity between an upcoming user's query and an FAQ entity can be enhanced because various users' queries contain rich syntactic information of the same FAQ entity as well as unexpected template subpaths created by registering users' queries. For example, consider the following two sentences:

S4: What is a searching task?

S5: How are searching tasks done?

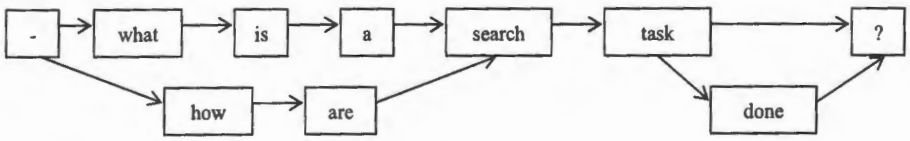
By representing these two sentences in template structure, these sentences can be matched and merged as shown in Figure 13 by using corresponding algorithms as described above. However, when the two subpaths are registered into template *T6*, an unexpected path, "*what is a search work done?*", is established at the same time. As a result, upcoming users' queries, such as "*what is a searching task done?*", "*what is a searching done?*" or "*what is searching*

assignment done?", can be covered by template *T6* in the future due to similar syntactic information already contained in the template.



Template T4 for the S4

Template T5 for the S5



Template T6 for the merging result of T4 and T5

Figure 13 the example of two templates merging

In summary, given a group of semantically related FAQ entities retrieved by the DLSI approach with term expansion technique, the hybrid approach can satisfy user expectations by using the template matching algorithm to reorder candidates in descending order of syntactic similarity. Moreover, the ability of template structure to evaluate syntactic similarity can benefit from the increase in the amount of subpaths by registering users' queries into corresponding FAQ templates.

5 General Introduction to an Implementation of the Hybrid Approach

TDT-FAQ Seeker, an implementation of the hybrid approach proposed above and named by the first letter of the main techniques (*T*erm expansion, *D*LSI, and *T*emplates) used in the hybrid approach, intends to provide a visual way for users to automatically set up an FAQ

retrieval system without domain limitations, retrieve few ordered FAQ entities which semantically and syntactically relate to a user's query written in format-free English from an FAQ collection, and improve the ranks of potential matched FAQ entities with higher syntactic and semantic similarity by gathering responses of users on the particular FAQ entities.

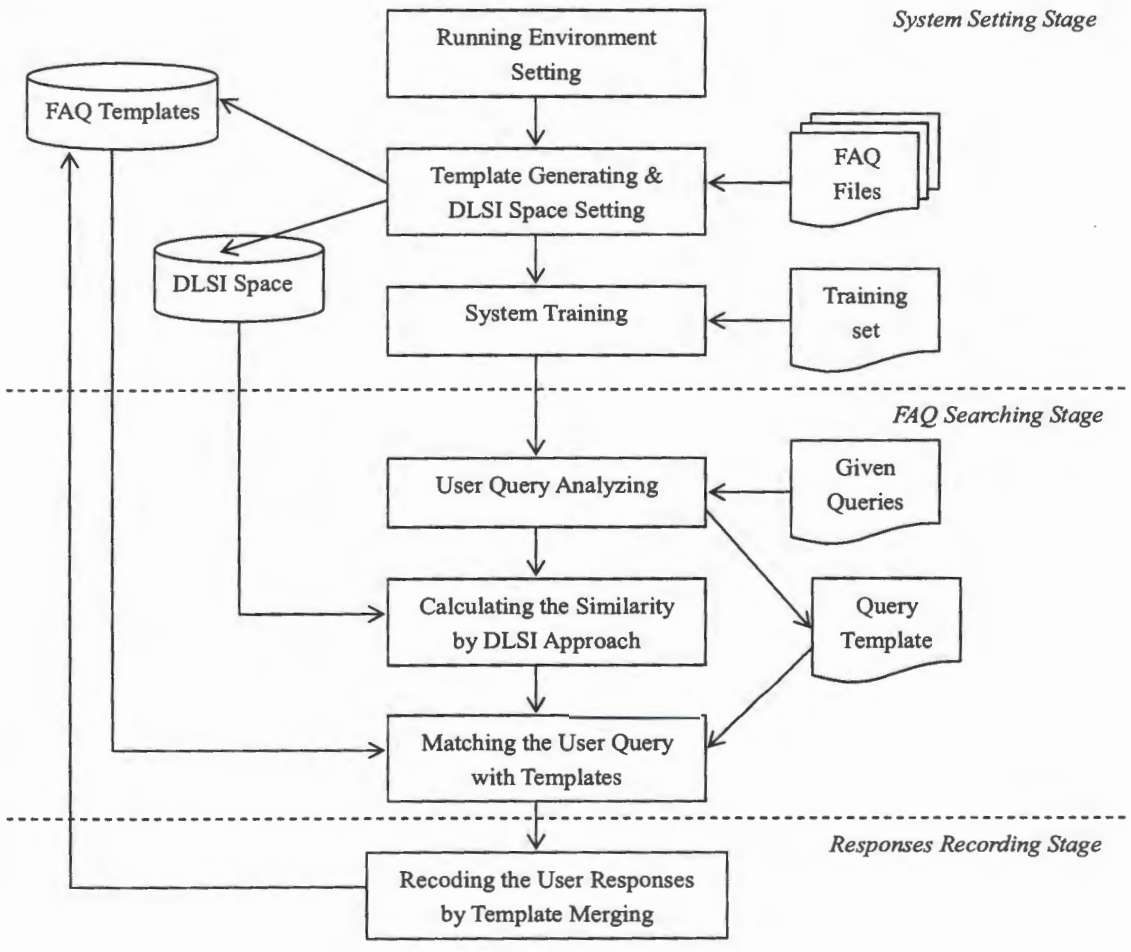


Figure 14: The Architecture of TDT-FAQ Seeker System

TDT-FAQ Seeker system is implemented in Java programming language (*JDK 1.4.2*)[3] on a Windows XP system with three open source code packages, JMAT[4], IGLU[5] and Grace[6], which are employed and modified for particular purposes. By following the framework shown in Figure 14, TDT-FAQ Seeker system contains three corresponding stages, System Setting, FAQ Searching and Response Recording for performing the FAQ retrieval task.

Considering an FAQ collection where each file stores an FAQ entity, the first stage, or *System Setting* stage, is designed to automatically generate the co-occurrence two-term pair list for expanding FAQ entities and users' queries, to construct differential term-vector spaces for semantic analysis by properly indexing and weighting terms, and to create templates of FAQ entities for further addressing syntactic similarity.

In the second stage of the hybrid approach, *FAQ searching*, and given a user query, the system applies the DLSI approach with the *term expansion* technique at the first step to retrieve a few semantically related FAQ entities, and then represents the user query and related FAQ entities in template structure for reordering semantically related FAQ entities in syntactic similarity by using the template matching method.

---

[3] http://java.sun.com/j2se/1.4.2/download.html

[4] a package provides the functions for matrix computing, http://jmat.sourceforge.net.

[5] a package provides the functions for standard IR methods, such Porter Stemmer and Stop-word filter, http://iglu-java.sourceforge.net

[6] a package provides the data structures for representing directed graphs and functions for its related algorithms, such as Topological Sort, http://www.gerwin-klein.de/grace.

In the last stage, or Responding Recoding stage, by tracking the actions of users linking their own questions to particular FAQ entities, the corresponding FAQ template can be expanded by using the template merging algorithm to register a user's query into a target template for enhancing the ability to catch syntactic similarity of an upcoming user's query and an FAQ entity.

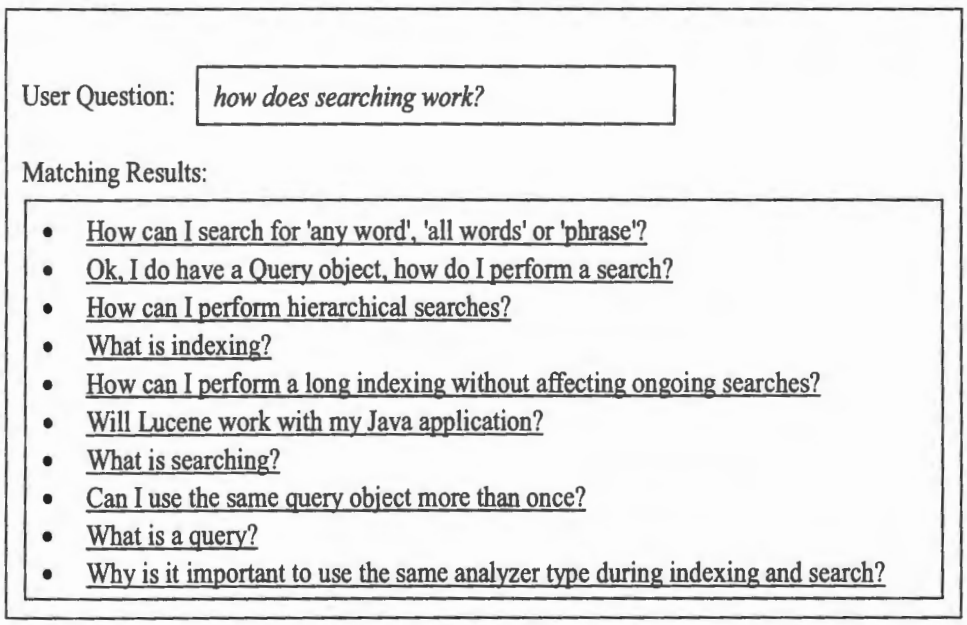


Figure 15: The Interface of the FAQ Retrieval System

For example, when a user enters the query, "*how does searching work?*", the system will return a candidate list, shown in Figure 15, containing FAQ entities which are not just semantically related to the user query but also syntactically similar with it. As a consequence, based on a user's particular bias, he or she can register what he or she asked into the template of a corresponding FAQ entity, such as "*what is searching*?", and examine the expanded

template visually, as shown in Figure 16.

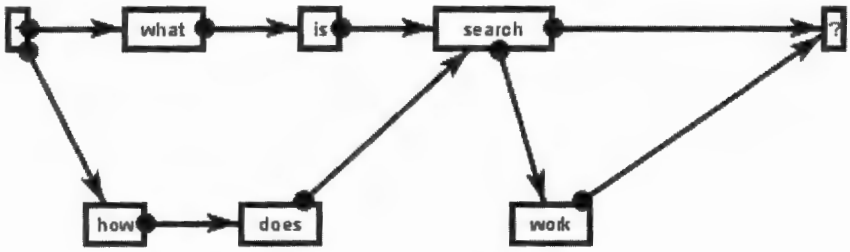


Figure 16: An Expanded Template

Chapter IV Evaluation of the Hybrid Approach

In this chapter, based on particular evaluation criteria, the results of a series of experiments are presented showing the value of the hybrid approach by comparing the performance of standard IR approaches for running FAQ retrieval tasks to the performance of the combination of original IR approaches and two additional methods, term expansion technique and template structure, with its matching and merging methods.

1 Evaluation Criteria

In order to answer a user's question, the proposed hybrid approach retrieves corresponding FAQs in the FAQ collection by combining three different techniques. Hence, in order to demonstrate the effectiveness of each technique in the new approach, as shown in Table 1, some approaches are grouped into four categories, keyword based, *tf-idf* based, LSI-based, and DLSI-based, and are implemented for comparing each approach to a corresponding and appropriate baseline.1) The baseline for the *tf-idf* approach, LSI approach, or DLSI approach is the keyword-based approach *Simple Keyword Matching*. 2) The baseline for the *tf-idf* approach with additional techniques is the standard *tf-idf* approach. 3) The baseline for the LSI approach with additional techniques is the standard LSI approach. 4) The baseline for the DLSI approach with additional techniques is the standard DLSI approach.

Category	Approach
Keyword Based Approach	Simple Keyword Matching
tf-idf Based Approaches	Standard *tf-idf*
	tf-idf + Term Expansion
	tf-idf + Template Matching
	tf-idf + Template Expansion + Template Matching
LSI Based Approaches	Standard LSI Approach
	LSI + Term Expansion
	LSI + Template Matching
	LSI + Template Expansion + Template Matching
DLSI Based Approaches	Standard DLSI Approach
	DLSI + Term Expansion
	DLSI + Template Matching
	DLSI + Template Expansion + Template Matching (The New Hybrid Approach)

Table 1: Some Approaches to Retrieve FAQ Entities

In addition, due to the ability of a template structure to improve efficiency in addressing syntactic similarity by appending various user queries into corresponding templates of FAQ entities, some experiments for proving this ability were conducted.

For evaluating the FAQ retrieval task, although the task is a special topic of IR, the traditional evaluation metrics, precision and recall for evaluating IR tasks are less suitable for evaluating the performance of the FAQ retrieval task than evaluating other IR tasks due to the following two reasons:

1) The efficiency of the two factors, precision and recall, is highly dependent on properly

setting relevant documents in a collection for each query belonging to a corresponding user query set. Since all entities in an FAQ collection focus on the detailed information of a general topic, FAQ entities are typically related to each other. The rules for properly distinguishing whether an FAQ entity semantically relates to a user query are notable obstructions for evaluating performance of FAQ retrieval tasks by the two factors, precision and recall.

2) As a practical problem in an FAQ retrieval task, users would rather see what they want at the top, rather than the end, of a candidate list. Hence, user satisfaction on retrieval results should be considered in any approach for solving practical problems such as FAQ retrieval. However, normal precision-recall measures do not give any indication of user satisfaction.

Hence, by assuming one and only one FAQ entity correctly matches each user query, a new evaluation measure for evaluating the FAQ retrieval task is designed using two independent metrics, retrieval precision and average rank. The retrieval precision metric is calculated by the number of users' queries which retrieve correctly matched FAQ entities over the total number of users' queries in the user query set. It estimates the probability of retrieving a correctly matched FAQ entity given a user query in a certain retrieval number. At the same time, the average rank metric is computed by the sum of the ranks of correct matched FAQ entities in the candidate list over the number of given users' queries which retrieve correctly matched FAQ items. The average rank represents the possible position where a correct

matched FAQ entity appears in a candidate list. By defining the two metrics, the approach with higher retrieval precision and lower average rank always has better performance than that demonstrated by an approach with lower retrieval precision, or an approach with the same retrieval precision but higher average rank.

Here, in order to easily learn the difference in retrieval precision between any two approaches, the factor, changes of precision, is defined as the difference of retrieval precision between any one of the approaches and corresponding baseline. A positive value means users have a greater chance of obtaining what they want with that approach. A negative value means the chance of obtaining the correct matching FAQ entity is lower. Similarly, in order to show the changes in user satisfaction level in an approach by comparing the corresponding baseline approach, the factor, or changes of average rank, is the ratio of difference of average rank between an approach and its baseline approach over the value of average rank of the baseline. A negative value means users have more opportunities to find what they want at the top of a candidate list; while a positive value means users have to take more time to scan the candidate list and locate what they want.

2 Evaluation of the Experimental Results of Different Approaches

Since no experimental results based on standard or published FAQ collection with a corresponding user query set is available for comparison, all approaches are evaluated by

using a particular FAQ collection: Lucene FAQs[1], with a corresponding user query set containing 521 users' queries (see detail in Appendix), each of which is composed by native speakers and each of which retrieves only one correctly matched FAQ entity.

2.1 Simple Keyword Matching

Supported by various efficient algorithms, keyword-based methods are widely used in solving IR problems and demonstrate an acceptable retrieval precision with a short response time. Hence, the keyword-based *Simple Keyword Matching* method for retrieving corresponding FAQ entities by regarding a user query is also evaluated.

Regardless of the complexity for searching related FAQ entities, a user's query is judged as matching an FAQ item if the number of matched non-stop-words in the FAQ entity exceeds a preset threshold value.

For an overview, the average number of retrieved FAQ entities is used to describe the performance of this approach. This metric is defined as the number of obtained matched FAQ entities over the total number of users' queries which could obtain matched FAQ entities. By using this metric, the amount of human effort required for searching the correct matched FAQs can be evaluated. By setting the preset threshold[2]. to 2, 3, and 4, the average numbers

[1] http://lucene.sourceforge.net/cgi-bin/faq/faqmanager.cgi

[2] If the number of non-stop-words in a user query is less than the threshold, then, we select the number of non-stop-words in the query as the threshold.

of retrieved FAQ entities are given in Table 2

Threshold	Correct Matched User Query		Average number of Retrieved FAQ Items
	Number	Precision	
≥ 2	351	67.37%	14.86
≥ 3	297	57.01%	6.11
≥ 4	231	44.34%	4.08
The number of non-stop-words used in user query	210	40.31%	3.98

Table 2: The Results of Simple Word Matching by Given a Threshold

For comparing this approach with the approaches of other categories in unified evaluation criteria, the retrieval precision and average rank are given by following the rules to rank matched FAQ entities in a candidate list with a certain number of matched FAQ entities as below.

(1) Given a user query, if the FAQ entity *A* has more matched non-stop-words than FAQ entities *B*, then order them as *A*, *B*.

(2) Given a user query, if the FAQ entity *A* has the same number of matched non-stop-words as FAQ entity *B*, and FAQ entity *A* has more matched stop-words than FAQ entity *B*, then order them as A, B.

(3) Otherwise, order them as *B*, *A*.

Similarly, by setting the retrieval number as 10, 15 or 20, the results for retrieval precision and average rank are as shown in Table 3.

Number of Returned FAQ Items	Results	
	retrieval precision	average rank
10	52.78%	5.97
15	59.11%	8.53
20	61.42%	10.61

Table 3: The Results of Simple Word Matching in Two Evaluation Metrics

2.2 Standard *tf-idf* Approach vs. Its Hybrid Methods

For evaluating the similarity of a user query and an FAQ entity, the standard *tf-idf* approach as a VSM approach represents each FAQ entity in a term vector by using *tf* and *idf* to weight any term appearing in a vector. The factor *tf* denotes the frequency of the term in a document. The factor *idf* stands for the inverse document frequency defined by $\log\left(\frac{N}{n}\right)$, where N is the total number of documents in the collection, n is the total number of documents which contain that particular word. By doing so, the standard *tf-idf* approach changes the problem of directly estimating similarity between a user's query and an FAQ entity into finding out how close the two weighted term frequency vectors are in a document space based on *cosine* measure:

$$sim(\vec{q},\vec{a}_i) = \frac{\vec{q}\cdot\vec{a}_i}{\|\vec{q}\|\cdot\|\vec{a}_i\|}$$

where $\vec{q}$ is the weighted term frequency vector of a user query, $\vec{a}_i$ is the weighted term frequency vector of i-th FAQ entity query.

tf-idf Method	Evaluation Factors	Retrieval Number		
		10	15	20
Standard	retrieval precision	75.05%	77.54%	79.27%
	changes of precision	22.27%	18.43%	17.85%
	average rank	1.85	2.27	2.86
	changes of average rank	-69.01%	-73.39%	-73.04%
Term Expansion	retrieval precision	76.39%	78.89%	81.38%
	changes of precision	1.34%	1.35%	2.11%
	average rank	2.93	3.67	4.31
	changes of average rank	58.38%	61.67%	50.70%
Template Matching	retrieval precision	75.05%	77.54%	79.27%
	changes of precision	0.00%	0.00%	0.00%
	average rank	1.64	1.96	2.36
	changes of average rank	-11.35%	-13.66%	-17.48%
Term Expansion & Template Matching	retrieval precision	76.39%	78.89%	81.38%
	changes of precision	1.34%	1.35%	2.11%
	average rank	1.71	2.08	2.43
	changes of average rank	-7.57%	-8.37%	-15.03%

Table 4: The Results of *tf-idf* Approach with Two Additional Techniques

With the standard *tf-idf* approach, the term expansion technique and template structure are combined with its matching algorithm either alone or together with the standard *tf-idf* approach. The standard *tf-idf* combined with the term expansion technique is designed to illustrate its contributions in enriching the lexical information of an FAQ entity or a user query by bridging the lexical distribution gaps. The standard *tf-idf* combined with template matching intends to advance the rank of correctly matched FAQ entities for reflecting what users want by addressing syntactic similarity of the user's query and a few FAQ entities

semantically related to that query.

By regarding the Simple Word Matching method as the baseline and setting the number of returning semantically related FAQ entities as 10, 15, or 20, performances of the standard approach and its combination methods are shown in Table 4.

2.3 Standard LSI Approach vs. Its Hybrid Approaches

As stated in Section 4.3 of Chapter II, by representing each document in a document collection as a normalized and weighted term-document vector based on the occurrences of each term across the whole document collection, the LSI approach as a more sophisticated VSM (vector space method) method analyzes semantic similarity between any user's query and each FAQ entity in the collection based on a lower dimensional term-document vector space where each dimension is meaning independent of the others.

Similarly, for showing the effectiveness of two additional methods, the term expansion technique and the template matching technique, these two methods alone or in a combination are combined with the standard LSI approach. Results of experiments on performance of the standard LSI approach and its hybrid approaches can be obtained with the most suitable values of k factor for constructing the semantic analysis space. The suitable value of k factor for each approach is decided by the best performance obtained by the approach with different possible values of k. The following experimental results are given over the baselines[3] with

[3] the baseline of the standard LSI approach is *Simple Word Matching*. The baseline of others is standard LSI approach.

the most suitable *k* factor for each approach in Table 5:

LSI Method	Evaluation Factors	Retrieval Number		
		10	15	20
Standard (*k=76*)	retrieval precision	77.74%	79.65%	81.00%
	changes of precision	24.96%	20.54%	19.58%
	average rank	2.15	2.47	2.85
	changes of average rank	-63.99%	-71.04%	-73.14%
Term Expansion (*k=48*)	retrieval precision	80.23%	81.19%	82.53%
	changes of precision	2.49%	1.54%	1.53%
	average rank	2.85	3.01	3.48
	changes of average rank	32.56%	21.86%	22.11%
Template Matching (*k=76*)	retrieval precision	77.74%	79.65%	81.00%
	changes of precision	0.00%	0.00%	0.00%
	average rank	1.83	2.17	2.42
	changes of average rank	-14.88%	-12.15%	-15.09%
Term Expansion & Template Matching (*k=48*)	retrieval precision	80.23%	81.19%	82.53%
	changes of precision	2.49%	1.54%	1.53%
	average Rank	1.85	2.22	2.46
	changes of average rank	-13.95%	-10.12%	-13.68%

Table 5: The Results of LSI Method with Its Hybrid Approaches

2.4 Standard DLSI Approach vs. Hybrid Approaches

As the core technique used in the proposed hybrid approach, the DLSI-based approaches distinguish themselves from other approaches based on VSM by using a more sophisticated model to address semantic similarity of any users' queries and one of the FAQ entities in a FAQ collection on two semantic analysis spaces which aim to describe the lexical difference

of two portions of each FAQ entity and the lexical difference between any two entities contained in an FAQ collection respectively.

DLSI Method	Evaluation Factors	Retrieval Number		
		10	15	20
Standard (*ki*=76,*ke*=10)	retrieval precision	80.42%	82.15%	85.80%
	changes of precision	27.64%	23.04%	24.38%
	average rank	2.56	2.73	2.95
	changes of average rank	-57.12%	-68.00%	-72.20%
Term Expansion (k*i*=48,*ke*=10)	retrieval precision	84.84%	86.77%	89.44%
	changes of precision	4.42%	4.62%	3.64%
	average rank	2.92	3.36	3.62
	changes of average rank	14.06%	23.08%	22.71%
Template Matching (*ki*=76,*ke*=10)	retrieval precision	80.42%	82.15%	85.80%
	changes of precision	0.00%	0.00%	0.00%
	average rank	1.73	1.98	2.21
	changes of average rank	-32.42%	-27.47%	-25.08%
Term Expansion & Template Matching (*ki*=48,*ke*=10)	retrieval precision	84.84%	86.77%	89.44%
	changes of precision	4.42%	4.62%	3.64%
	average rank	1.76	2.07	2.41
	changes of average rank	-31.25%	-24.18%	-18.31%

Table 6: The Results of DLSI Method with Its Hybrid Approaches

Experiments illustrating performance of the DLSI approach in retrieving FAQs and performance of this approach combined with the term expansion technique and template matching technique separately or together are conducted by steps described in Chapter III. Similarly, by choosing the best k factors, k_i and k_e, for defining the k-interior differential space and k-exterior differential space in DLSI-based approaches, experimental results are

given over the baselines[4] as shown in Table 6.

2.5 Suggestions Derived From Learning the Experimental Results

First, regarding results shown in Table 2, when criterion for judging whether an FAQ entity matches a given user query become more strict, the chances of a user fetching a correctly matched FAQ entity decreases. In other words, as with other IR tasks, performance of an FAQ retrieval task also faces the difficulty of discovering similarity by matching the same words appearing in limited context directly.

Second, upon evaluation of results shown in Table 3, Table 4, Table 5 and Table 6 based on the two metrics, retrieval precision and average rank, the following is suggested:

(1) Approaches based on the vector space model, such as *tf-idf* based, LSI-based and DLSI-based, demonstrate enhanced retrieval performance in comparison with the keyword-based approach, or Simple Word Matching method. Since the approaches based on VSM use term occurrence in each single document and document collection to extract relationships among terms and documents across the whole FAQ collection (for avoiding lexical information scarcity in a single FAQ entity), VSM-based approaches show the ability to offer users greater chances to correctly retrieve a matched FAQ entity, and to do so with much less human effort.

[4] For the standard DLSI and combination approaches, the baseline is LSI approach.

(2) Due to the restricted usage of vocabulary in each FAQ entity, when the overlapped information or additional representations of FAQ entities are used, the standard DLSI method exhibits much more retrieval precision than the standard *tf-idf* approaches or the standard LSI approach. Comparing experimental results of retrieval precision performed by the standard DLSI approach with results performed by other two standard approaches, it improves precision by 5.5% and 3.33% on average with different returned candidates over the standard *tf-idf* and standard LSI approaches respectively. Hence, improvement in retrieval precision suggests that the DLSI approach is more suitable for solving the FAQ retrieval task.

(3) Across all results shown in corresponding tables, when VSM based approaches include the term expansion technique as a built-in method capable of bridging lexical distribution gaps between a user query and an FAQ entity and between two portions of any one FAQ entity, these hybrid approaches retrieve more matched FAQ entities than using the standard version of VSM approaches alone. Since the term expansion method involves appending additional terms, each of which could be either helpful or harmful for representing characteristics of the meaning of particular contents, ranks of correctly matched FAQ entities in the returned candidate list are, to some degree, disordered by VSM based approaches with the term expansion technique.

(4) By combining the template matching method with three baseline approaches (standard *tf-idf* approach, standard LSI approach, and standard DLSI approach), all these hybrid

approaches show an ability to increase ranks of correctly matched FAQ entities compared with original baseline approaches. By comparing values of the average rank of each standard approach and the combination of it and the template matching method with different returned candidates, average improvement in the average rank of the *tf-df* approach, LSI approach and DLSI approach amounts to 15.02%, 14.06%, and 28.36%, respectively. Since the template matching method aims to improve ranks of syntactically matched FAQ entities by reordering semantically related FAQ entities in a candidate list, the template matching method does not change any retrieval precision of the three baseline approaches.

(5) The combined use of both term expansion and template matching techniques is capable of providing a noticeable improvement in retrieval precision and average ranks of correctly matched FAQ entities for the VSM-based approaches. Term expansion enriches vocabulary in each single FAQ entity a user query thus giving users more chances to get correctly matched FAQ entities. Template structure with its matching technique improves the ranks of semantically related FAQ entities and users' queries by evaluating their syntactic similarities.

3 Evaluation of the Experimental Results of DTE

Since template structure plays an important role in meeting user satisfaction by using the template matching algorithm to address syntactic similarity between any two templates and applying the template merging algorithm to register one template into another, the new

hybrid approach proposed in this thesis expands corresponding templates of FAQ entities to further meet user satisfaction by following the dynamic template expansion procedure.

3.1 Merging Rules for Simulating User Actions

In the real world, human users always have chances to choose or miss the correctly matched FAQs in the list and to register what they queried due to various reasons, such as form of the semantically related FAQs being dissimilar to users' inputs. In order to simplify users' actions in the real world here, by supposing that users can always properly judge whether or not an FAQ entity in a candidate list correctly matches their query and two rules are followed to merge the their queries into the templates of corresponding correctly matched FAQs, a series of experiments based on a simple case, the DTE of a particular FAQ entity, and a complex case, the DTE of all FAQ entities in the collection, were set up. The two merging rules are Worst Rank First (WRF) and Best Rank First (BRF). WRF is defined as always merging a user's query into a correctly matched FAQ entity's template, which give the worst (the largest in numeric value) rank among the ranks of all correctly matched FAQ entities given by other users' queries. BRF is defined as always merging a user's query into the template of a correctly matched FAQ entity which has the best (the smallest in numeric value) rank among the ranks of all correctly matched FAQ entities given by other users' queries. WRF depicts the normal action of a user who has enough confidence to choose an FAQ entity with distinct syntactic difference between a user query and itself as the correctly matched FAQ entity by registering what he or she wants into the template of the correctly matched FAQ entity.

Similarly, BRF depicts the normal action of a user who always chooses an FAQ entity with relative higher syntactic similarity between a user query and itself as the correctly matched FAQ entity by merging his or her query into the template of the FAQ entity.

These two merging rules reflect a perfect world. Unfortunately, the real word is filled with uncertainties and hard to simulate. The aim of demonstrating these experiments is to give a general idea of how the DTE works.

3.2 Experiments on DTE with a Simple Case

NO.	User Query	INI	R1	R2	R3	R4
Q1	How does searching work?	4	4	1	1	1
Q2	How is a search done?	7	7	1	1	1
Q3	What are the processes behind searching?	4	4	4	1	1
Q4	What are the mechanisms behind searching?	5	5	5	1	1
Q5	What are the operations behind searching?	3	3	3	1	1
Q6	What are the means behind searching?	6	6	6	1	1
Q7	What are the procedures behind searching?	5	5	5	1	1
Q8	What does searching involve?	2	2	2	2	1
Q9	What is the input for a search?	1	1	1	1	1
Q10	Are the hits listed in order or importance?	1	1	1	1	1
Q11	Can you give me information about searching?	8	1	1	1	1
Average Rank of the Correct Matching FAQ Item		4.18	3.55	2.73	1.09	1.00

The User Question of which the background color is gray is registered into template in each round

Table 7: A Sample of DTE based on Worst Rank First

To show rank improvements gained by DTE, two experiments were set up by following merging rules WRF and BRF to merge 11 users' queries into the template of a particular FAQ entity "*What is searching*?"

Assuming only one user query is appended into the template of an FAQ entity by following the WRF in each round, rank of the corresponding correctly matched FAQ entity given by any one of the 11 users' queries becomes 1 by merging 4 users' queries into the template of the FAQ entity in a sequence, as shown in Table 7. The expanded template of the FAQ entity after registering four users' queries into the template is shown in Figure 17.

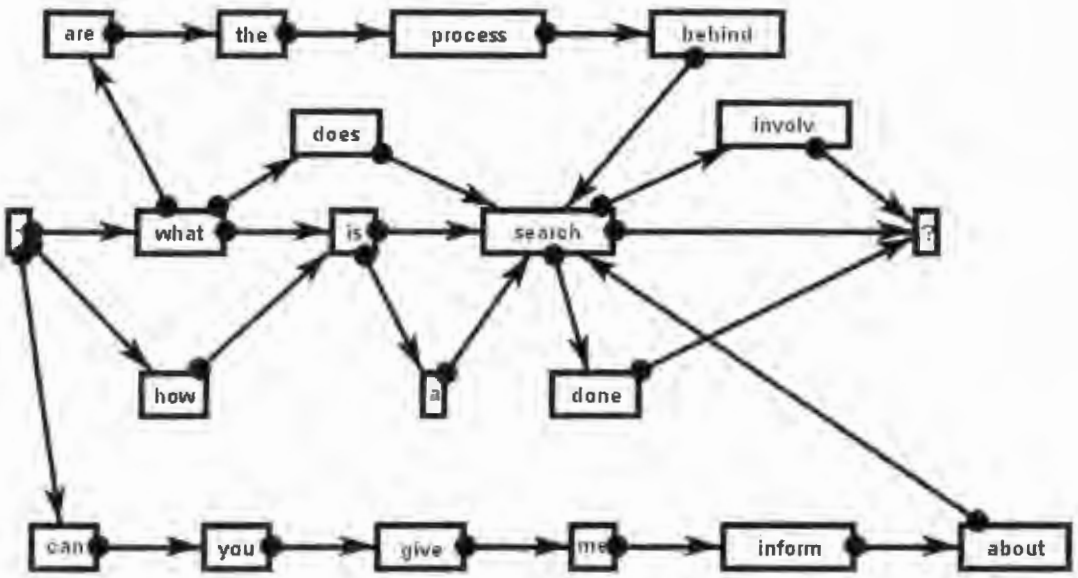


Figure 17: An Expanded Template Show in DAG Form

Similarly, by supposing only one of the users' queries are appended into the template of the FAQ entity by following the BRF in each round, rank of the corresponding correctly matched FAQ entity given by any one of 11 users' queries becomes 1 by registering 11 users' queries into the template of the FAQ entity, as shown in the sequence of Table 8. Due to the complicated DAG generated by registering users' queries one by one, the template in DAG

form is not provided here.

NO.	INI	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11
Q1	4	4	4	4	4	4	4	4	1	1	1	1
Q2	7	3	3	3	3	3	3	3	3	3	1	1
Q3	4	1	1	1	1	1	1	1	1	1	1	1
Q4	5	2	1	1	1	1	1	1	1	1	1	1
Q5	3	1	1	1	1	1	1	1	1	1	1	1
Q6	6	1	1	1	1	1	1	1	1	1	1	1
Q7	5	1	1	1	1	1	1	1	1	1	1	1
Q8	2	2	2	2	2	2	2	2	1	1	1	1
Q9	1	1	1	1	1	1	1	1	1	1	1	1
Q10	1	1	1	1	1	1	1	1	1	1	1	1
Q11	8	8	8	8	8	8	8	8	8	8	8	1
Average Rank	4.18	2.27	2.18	2.18	2.18	2.18	2.18	2.18	1.82	1.82	1.64	1.00

The User Question of which the background color is gray is registered into template in each round

Table 8: A Sample of DTE based on Best Rank First

3.3 Experiments on DTE with a Complicated Case

The template matching method in the proposed hybrid approach is employed as a filter for choosing the syntactically related FAQ entities from semantically related candidates returned by the DLSI approach. By obtaining a candidate list which contains more semantically related FAQ entities chosen by the DLSI approach, the hybrid approach offers users more opportunities to obtain correctly matched candidates with better ranks in the same number of returned candidates.

By setting the number of semantically related candidates returned by the DLSI approach to 15, and the number of candidates filtered by the template matching method to 10, the

experimental results[5] evaluated on the two metrics, retrieval precision and average rank, are given by the two merging rules WRF and BRF, separately, in Tables 9 and 10, and Figures 18 through 21, inclusive

(1) Experimental results by following the WRF

Round	Retrieval Precision	Changes of Precision	Average Rank	Changes of Average Rank
1	85.22%	0.00%	1.69	0.00%
5	85.60%	0.38%	1.63	-3.55%
10	86.56%	1.34%	1.5	-11.24%
25	86.56%	1.34%	1.29	-23.67%
50	86.56%	1.34%	1.12	-33.73%
75	86.56%	1.34%	1.05	-37.87%
97	86.56%	1.34%	1	-40.83%

Table 9: The Sample Rounds of the Complicated Example based on WRF

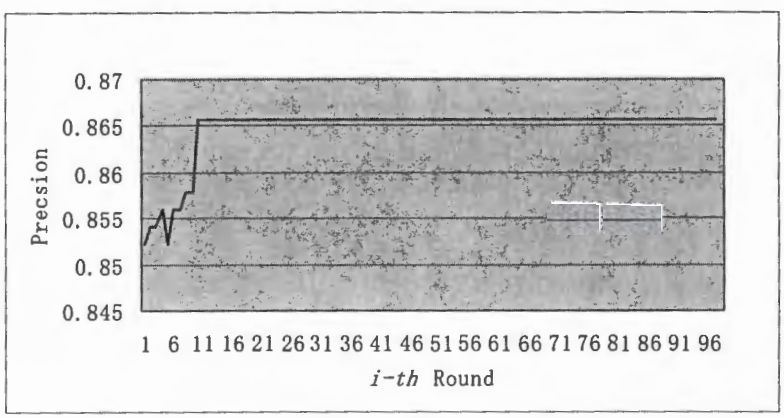


Figure 18: The Variation of Retrieval Precision of the Complicated Example based on WRF

[5] The baseline of each round is the first round.

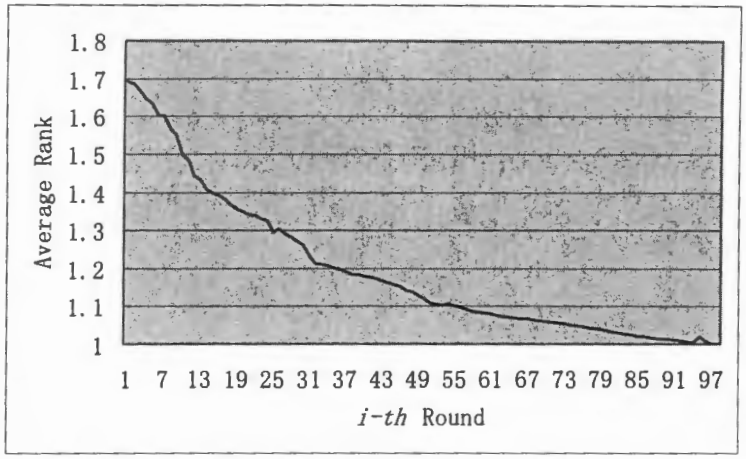


Figure 19: The Variation of Average Rank of the Complicated Sample based on WRF

(2) Experimental results by following BRF

Round	Retrieval Precision	Changes of Precision	Average Rank	Changes of Average Rank
1	85.22%	0.00%	1.8423	0.00%
5	85.60%	0.38%	1.8445	0.12%
10	85.60%	0.38%	1.8445	0.12%
25	85.03%	-0.19%	1.8262	-0.87%
50	85.03%	-0.19%	1.8104	-1.73%
75	85.03%	-0.19%	1.7788	-3.45%
97	85.03%	-0.19%	1.7607	-4.43%
130	85.22%	0.00%	1.7432	-5.38%
140	85.41%	0.19%	1.7393	-5.59%
160	85.60%	0.38%	1.7174	-6.78%
200	85.60%	0.38%	1.7242	-6.41%
300	85.60%	0.38%	1.574	-14.56%
400	86.18%	0.96%	1.3608	-26.14%
451	86.56%	1.34%	1	-45.72%

Table 10: The Sample Rounds of the Complicated Sample based on BRF

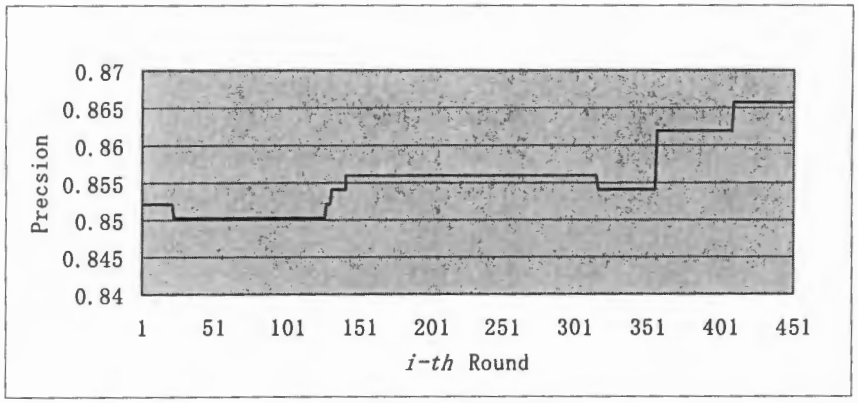


Figure 20: The Variation of Retrieval Precision of the Complicated Sample based on BRF

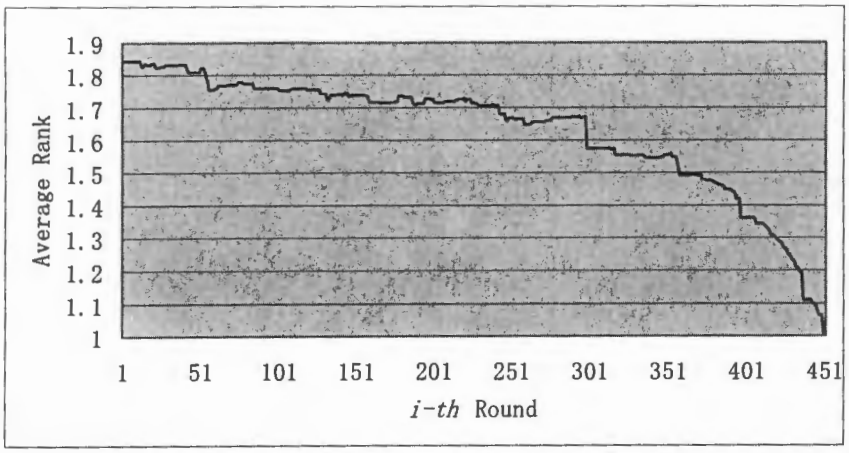


Figure 21: The Variation of Average Rank of the Complicated Sample based on BRF

3.4 Conclusions Derived from Experimental Results

As shown in a simple example, the average rank of the correctly matched FAQ entity is gradually reduced by increasing the number of users' queries into the template based on WRF and BRF merging rules. In other words, DTE enhances the ability to address syntactic similarity between a user's query and the template when syntactic information contained in users' queries on the same topic is appended into the templates.

Second, as shown by results of DTE experiments in the complicated example, the hybrid approach demonstrates an improvement in precision of retrieving corresponding FAQ entities and an increase in ranks of the correctly matched FAQ entities in the returned candidate list by the WBF and RBF merging rules. Whether or not the merging rules are used, there are benefits with DTE for obtaining more opportunities for a correctly matched FAQ entity with better rank in a narrower searching space. When syntactic information is registered into corresponding templates more and more, retrieval precision is not improved since the characteristics of templates are already covered by enough syntactic information.

Third, by regarding rank of the corresponding correctly matched FAQ entity ranked as 1 by taking 4 rounds with WRF and 11 rounds with BRF in the simple example, or by taking 97 rounds with WRF and 451 rounds with BRF in the complicated case, the amount of distinctive syntactic information contained in users' queries to retrieve correctly matched FAQ entities with worse rank enriches the syntactic information contained in the template much more than syntactic information contained in users' queries which retrieve correctly matched FAQ entities with better rank.

Finally, retrieval precision can be made worse by appending several users' queries, such as in the 6-*th* round in Figure 18 and in the 22-*th* round in Figure 20. When the sequences of terms and stop-words widely appearing in related FAQ entities are registered into corresponding templates, more semantically related candidates are ranked as the best matched FAQ entities rather than improving the correctly matched FAQ entity only. Thus, retrieval precision of

correctly matched FAQ entities is somewhat degraded since common syntactic information contributes much in distinguishing one template from another. When more specific syntactical information on particular FAQ entities is introduced into corresponding templates, retrieval precision is raised in the long run. Similarly, variations in the average rank of correctly matched FAQ entities also demonstrate congruous phenomena, as shown in Figure 19 and Figure 21. When common vocabularies organized in the same order are registered into FAQ templates, differences among templates are sometimes hardly distinguished by the template matching algorithm due to obtaining their same score and distance of LCSs at the same time. However, when specific vocabularies and vocabulary orders are appended into FAQ templates, particular characteristics of each template can be described.

Chapter V Conclusion

This chapter summarizes this thesis as well as discusses further possible research on this topic.

1 Summary

In this thesis, an innovative approach combining the DLSI approach with two additional techniques, term expansion technique and template structure, is introduced for solving the FAQ retrieval task. The hybrid approach proposed aims to automatically retrieve few FAQ entities that are semantically as well as syntactically similar to a user query written in English by using statistical methods to analyze semantic similarity based on lexical distribution in unified document vector spaces, and by further representing FAQ entities and users' queries in a particular information organization form, or template, to address syntactical similarities between a user query and each candidate semantically related to that user query.

A series of experiments formulated to demonstrate efficiency of each technique used in the hybrid approach are presented. Advantages of the hybrid approach for the FAQ retrieval task are confirmed by comparing performances of standard IR approaches with performances of various combinations of IR approaches and the two additional techniques of term expansion technique and template structure

2 Further Research

As shown in experiment results, the precision of retrieving correctly matched FAQ entities

can either be improved or made worse because the term expansion technique is directly related to appending extra terms, which are either helpful or harmful for representing a particular FAQ entity, into the corresponding portions of FAQ entities or users' queries. By regarding the portions of any FAQ entity as a sentence in the current research and implementation stage, a preset threshold is used to pick up terms with higher co-occurrence probability of the corresponding term-sentence pair for expanding portions of an FAQ entity. As in the similar IR technique known as query expansion [HPP+95, XC98], the threshold applied in the term expansion technique was chosen based on experimental results in a particular document collection (the collection of Lucene FAQs). Although the experiments conducted in Chapter IV demonstrate feasibility of the proposed method, a method for automatically judging which extra terms definitely contribute to distinguishing an FAQ entity from other FAQ entities in different FAQ collections is required.

In the hybrid approach, the template structure is well suited to addressing syntactic similarity between any two templates by matching various subpaths contained in them. With the rules applied in the hybrid approach, each template is created and expanded by following original word order of terms and stop-words shown in a sentence of any FAQ entity or any user query. However, syntactic information, such as word order of phrases or syntactic categories to which each word belongs in a sentence, that is contained in a sentence is far beyond what is used now. Hence, developing a parser for constructing a template in valid grammar rules based on natural language processing techniques becomes an interesting topic for more

precisely representing characteristics of any FAQ entity and matching any two of templates.

Finally, considering there is no standard FAQ collection with corresponding user query set for evaluating the FAQ retrieval task, composing a well organized FAQ collection and corresponding user query set is necessary for further research on this topic.

Appendix: User Queries Set of the FAQ Collection of Lucene

faq1: What are Lucene's features?
faq1: What language is Lucene written in?
faq1: When was Lucene started?
faq1: Who started Lucene?
faq1: What is Jakarta?
faq1: How fast is Lucene?
faq1: What is the size of the index?
faq1: What are the features of the search engine?
faq1: What does the API allow me to do?
faq1: When did Lucene become a Jakarta project?
faq2: Where is Lucene's website?
faq2: What is Lucene's URL?
faq2: Where is the Apache Jakarta website?
faq3: Who created Lucene?
faq3: Who is Doug Cutting?
faq3: What did Doug Cutting do?
faq3: Where can I find information about Doug Cutting?
faq4: Where is this FAQ located?
faq4: Where is the latest version of this FAQ located?
faq5: Who can I contact to help administer this FAQ?
faq5: Can I help administer this FAQ?
faq6: Where can I find another Lucene FAQ?
faq6: Is there another Lucene FAQ?
faq6: Where is the jGuru Lucene FAQ?
faq7: Can I download Lucene?
faq8: Which programs will Lucene work with?
faq8: What kind of applications will Lucene work with?
faq8: Can Lucene search e-mail archives?
faq8: Can Lucene search online documentations?
faq8: Can Lucene search visited web pages?
faq8: Can Lucene search websites?
faq8: Can Lucene search collections of documents?
faq9: Is anyone working for the Lucene?
faq9: Can I be a developing member of Lucene project?
faq9: Can I do something for Lucene?
faq10: Is Lucene free?
faq10: Which license is Lucene governed by?

faq10: Where can I find information about ASL?
faq10: Where can I find information about Apache Software License?
faq11: What is LGPL?
faq11: Is Lucene ruled by the Apache Software License?
faq11: Is Lucene governed by the Apache Software License?
faq11: Is Lucene regulated by the Apache Software License?
faq12: How do I use Lucene?
faq12: Where can I find the Lucene mailing list?
faq12: Is there a Lucene mailing list?
faq12: Where can I find more information about Lucene?
faq12: Where I can learn how to use Lucene?
faq13: What else can I use besides Lucene?
faq13: Can I suggest an alternative to Lucene?
faq14: What do I need to run Lucene?
faq14: Which systems does Lucene work on?
faq15: Is Lucene a self-contained library?
faq15: Do I need a third party database to use with Lucene?
faq16: Does Lucene have a web crawler?
faq16: Do I have to locate the the documents that Lucene searches through?
faq17: Can I decide which documents to store in the index?
faq17: Can I decide what Lucene stores in the index?
faq18: Where can I join the Lucene mailing list?
faq18: How can I contact the developers?
faq18: How can I contribute to Lucene?
faq18: How can I help develop Lucene?
faq19: What is an index?
faq19: What can I store in an index?
faq19: What can Lucene store in an index?
faq19: What kind of indices does Lucene create?
faq19: What kind of indexes does Lucene create?
faq20: Where does Lucene store the index?
faq20: Where does Lucene put the index?
faq20: What if I use multiple independent indices?
faq20: How does Lucene store the index database?
faq20: Can Lucene provide in-memory storage of the index?
faq20: Can Lucene map data to a third party database?
faq21: What is the IndexWriter?
faq21: How do I use the IndexWriter?
faq21: How do I make an index of a set of documents?
faq21: How do I make an index?
faq22: How do I use an incremental update of the index?

faq22: How do I update the index incrementally?
faq22: What are the ways of updating the index?
faq22: How do I update the index?
faq22: What is the best way to update the index?
faq23: How can I remove documents from an index?
faq23: How do I delete documents with the IndexReader?
faq23: What are the methods of removing documents from an index?
faq23: What does delete(int) do?
faq23: What does delete(Term) do?
faq24: How do I use the addDocument() method?
faq25: What is the purpose of the Document objects?
faq25: What is the purpose of the Document class?
faq26: What are the attributes of a Field object?
faq26: What does a Field object do?
faq26: How are the documents and hit information represented?
faq26: What are the types of fields?
faq27: Which field type should I use?
faq27: How to I choose a field type?
faq28: Can Lucene access external documents?
faq29: Can Lucene extract the content and links of HTML or other document formats?
faq29: How can I extract the content and links of HTML or other document formats?
faq29: Does Lucene include an HTML parser?
faq29: Where can I find an HTML parser?
faq30: Can Lucene extract the content of PDF, Word, or other document formats?
faq30: Do I need to provide a parser or extractor for every document type I want to index?
faq31: What do Analyzers do?
faq31: How is the content of the document broken into terms?
faq31: How is the content of the document broken into tokens?
faq32: Can the same Analyzer be used more than once?
faq32: Can I use the same Analyzer more than once?
faq33: Should I use the same analyzer during indexing and searching?
faq33: What if I use different analyzer types during indexing and searching?
faq34: Are there any recommended analyzers?
faq34: Which Analzyer should I choose?
faq35: How do I write my own customer analyzer?
faq35: Can I see a sample customized analyzer?
faq36: How can I add functionality that is not in Lucene's token filters?
faq37: Is the order of the token filters used by an analyzer important?
faq37: Should I consider the order of the token filters used by an analyzer?
faq38: What do I need to consider when using a filter that uses a word dictionary?
faq38: How do I use a filter that uses a word dictionary?

faq39: Can a filter generate many tokens for a single input token?
faq39: How can I associate one input token with multiple tokens?
faq39: Can a filter generate multiple tokens for one input token?
faq40: How can I observe the effect of an analyzer on a given text?
faq40: How can I see the effect of an analyzer on a text?
faq40: How can I tell if my analyzer is effective?
faq40: What do I use to create a TokenSteam over a string?
faq41: What is the PorterStemmer?
faq41: What does the PorterStemmer do?
faq41: What is the Porter Stemming algorithm?
faq41: Where can I find more information about the Porter algorithm?
faq41: Can I see a demonstration of the Porter algorithm?
faq41: Does the PorterStemmer apply to other languages?
faq42: How can I use index optimization?
faq42: How can I optimize the index?
faq42: How can I compact the index database?
faq42: How can I speed up queries?
faq43: When are new segments created?
faq43: How do I reduce the number of segments?
faq43: What effect does index optimization have on segments?
faq44: How can I make queries match synonyms?
faq44: How can I use term aliasing?
faq44: Does Lucene provide a tokenizer that supports term aliasing?
faq45: What does the stop filter do?
faq45: How do the token filters remove from the indexed text words?
faq45: How can I remove words from the indexed text?
faq45: Which words should I use the Stop Filter for?
faq45: Which class uses the Stop Filter?
faq46: Can I use Lucene for documents in other languages?
faq46: Which languages does Lucene support?
faq47: What is the speed of Lucene's indexing?
faq48: How do I perform a long indexing when search is in progress?
faq48: How do I perform indexing when someone is searching?
faq48: Does editing the index affect ongoing searches?
faq48: What if I edit the index and someone is searching at the same time?
faq49: Is Lucene cross platform?
faq49: Which platforms does Lucene work on?
faq50: Does Lucene remove all old index files when I recreate the index?
faq50: Are the old index files automatically deleted when I rewrite the index?
faq51: How does searching work?
faq51: How is a search done?

faq51: What are the processes behind searching?
faq51: What are the mechanisms behind searching?
faq51: What are the operations behind searching?
faq51: What are the means behind searching?
faq51: What are the procedures behind searching?
faq51: What does searching involve?
faq51: What is the input for a search?
faq51: Are the hits listed in order or importance?
faq51: Can you give me information about searching?
faq52: How does Lucene represent queries?
faq52: How does Lucene represent a query?
faq52: What is a query matched to?
faq52: What is a query compared with?
faq53: How can I make a query object represent a certain query?
faq53: How can I construct a query object?
faq53: How can I obtain a query object?
faq53: How can I convert a query string to a query object?
faq53: What are the ways in which I can parse a query?
faq53: Do I have to pass the query parameters through the same analyzer that was used to index the document?
faq54: What is the definition of "terms"?
faq54: What are the basic units for indexing and searching?
faq54: What is tokenizing?
faq54: How is a string tokenized?
faq54: What is the process of breaking a string to its terms?
faq54: How does Lucene different from "grep"?
faq54: Are queries values case sensitive?
faq54: Are queries values case insensitive?
faq54: Are term values case sensitive?
faq54: Are term values case insensitive?
faq54: How can I make the search non case sensitive?
faq55: What is the syntax of the query parser?
faq55: What is the syntax of the query strings?
faq55: Are queries case sensitive?
faq55: Are terms case sensitive?
faq56: What are term queries represented by?
faq56: How does a boost factor affect a term query?
faq57: What can I use to increase or decrease the ranking of the hits?
faq57: What can I use to change the ranking of documents that match a query?
faq57: When should I use a boost factor?
faq57: How can I make closer matches displayed first?

faq57: How can I rank the importance of the hits?
faq58: What is a phrase query matched against?
faq58: Is the order of words important in a phrase query?
faq58: What class is a phrase query represented by?
faq58: Can a phrase query have a boost factor?
faq58: How does the slop factor affect a phrase query?
faq59: What are "and", "or", and "not" called?
faq59: What are Boolean queries represented by?
faq59: What are sub queries?
faq59: What are the qualifiers of sub queries?
faq59: How are Boolean queries constructed?
faq59: Can I see an example of a Boolean Query?
faq60: Can I use query objects multiple/many times?
faq60: Is it safe to reuse a query object?
faq60: When can I reuse a query object?
faq60: How can I use a query object more than once?
faq61: How can I tell if the query structure I created is correct?
faq61: How can I tell if the query structure I created is acceptable?
faq61: How can I know if the query structure I created is appropriate?
faq61: How can I know if the query structure I created is valid?
faq61: How can I diagnose related code?
faq61: How can I test related code?
faq61: How can I check related code?
faq61: How can I validate related code?
faq61: How can I query related code?
faq62: Do I have to replicate a query for documents with multiple fields?
faq62: How can I avoid writing a query many times for multiple fields?
faq62: Do matches in longer documents result in lower ranking?
faq63: How can I make Lucene make partial matches or find matches for prefixes expressions?
faq63: How can I make Lucene make partial matches or find matches for suffixes expressions?
faq63: How can I make Lucene make partial matches or find matches for regular expressions?
faq63: Does Lucene support general regular suppressions?
faq63: Does Lucene allow general regular suppressions?
faq63: Does Lucene support word prefixes?
faq63: Does Lucene allow word prefixes?
faq63: Can Lucene find matches using word prefixes expressions?
faq63: Can Lucene find matches using word suffixes expressions?
faq63: Can Lucene find matches using word regular expressions?

faq63: Can Lucene search prefixes expressions?
faq63: Can Lucene search suffixes expressions?
faq63: Can Lucene search regular expressions?
faq64: How do I perform a search?
faq64: What do I do after indexing the documents and making a query object?
faq65: What information does Lucene provide for each hit?
faq65: How do I access the hits?
faq65: How do I obtain the hits?
faq65: How do I access the hit scores?
faq65: How do I obtain the hit scores?
faq65: What is a hit score?
faq65: How is a document ordered with respect to hit scores?
faq66: How do I filter the hits?
faq66: How do I filter the search results?
faq66: What is a search query?
faq66: What is a selective collection?
faq67: How can I limit the number of hits?
faq67: How can I limit the number of search results?
faq67: Can Lucene restrict the number of hits?
faq67: Can Lucene limit the number of results?
faq67: Can Lucene restrict the number of results?
faq67: Can Lucene limit the number of hits?
faq68: How can I change the starting position of the hit list?
faq68: How can I change the starting page of the hit list?
faq68: How can I limit hit list range?
faq68: How can I restrict hit list range?
faq69: How can I limit the date range of the hit list?
faq69: How can I restrict the date range of the hit list?
faq69: How can I filter hits of a specific date range?
faq69: How do I make Lucene return hits only from certain dates?
faq69: How do I use the DateFilter?
faq70: How does Lucene encode date values?
faq70: Can I use more than one date value?
faq70: Can I have more than one date field?
faq71: Can I insert date values in regular text fields?
faq71: Can I include date values in regular text fields?
faq71: Can I enter date values in regular text fields?
faq71: Can I incorporate date values in regular text fields?
faq72: How do I page the hit list?
faq72: What are the ways in which the hits of each page are listed?
faq72: How can I separate the results into pages?

faq72: How can I separate the matches into pages?
faq72: How can I separate the hits into pages?
faq72: What are repeating searches?
faq72: How do I use repeating searches?
faq72: What are cached searches?
faq72: How do I perform cached searches?
faq73: Will the same query return the same hits each time?
faq73: Will the same query return the same results each time?
faq73: Are the search hits deterministic?
faq73: Are the search matches deterministic?
faq73: Are the search results deterministic?
faq74: How do I rate each hit with stars?
faq74: How do I score each hit with stars?
faq74: How do I rate the results with stars?
faq74: How do I score the results with stars?
faq74: How do I convert the hits score to a star rating?
faq74: How do I convert the hits ranking to a star rating?
faq75: Is the search method case sensitive?
faq75: Are the searches case insensitive?
faq76: What are the possible causes of not being able to find matches for a query?
faq76: Why can't I find something?
faq76: What are the reasons that cause a search to fail?
faq76: Why will a search not return any results?
faq77: How will searches with numbers be affected?
faq77: How will searches with symbols be affected?
faq77: How will searches with special characters be affected?
faq77: How does Lucene search numbers?
faq77: How does Lucene search symbols?
faq77: How does Lucene search special characters?
faq77: How are numbers treated?
faq77: How are symbols treated?
faq77: How are special characters treated?
faq78: How can I make the search match any word of a query?
faq78: How can I make the search match all words of a query?
faq78: How can I make the search match phrase of a query?
faq78: How can I return the hits that match any word, all words, or phrase?
faq78: How can I return the results that match any word, all words, or phrase?
faq78: How do I select the number of words that are used in searching?
faq78: How do I select the number of words that are used in matching?
faq78: How do I select which words to use in searching?
faq78: How do I select which words to use in matching?

faq78: How do I choose which words to use in searching?
faq78: How do I choose which words to use in matching?
faq79: Will a search for a word match the conjugated forms of the word?
faq79: Will a search for a word match the root forms of the word?
faq79: Will Lucene find hits that are related to the query?
faq79: Will Lucene find matches that are related to the query?
faq79: Will Lucene find words that are related to the query?
faq79: Will the results include conjugated forms of a word?
faq79: Will the results include conjugated forms of a query?
faq79: How do I use the root forms of a word in searches?
faq79: How do I make the Lucene show results that match a conjugated form of the query?
faq79: How do I make the Lucene show results that match a conjugated form of the term?
faq79: How do I make the Lucene show that match a conjugated form of the word?
faq79: How do I make the Lucene show results that match a root form of the query?
faq79: How do I make the Lucene show results that match a root form of the term?
faq79: How do I make the Lucene show that match a root form of the word?
faq79: How do I make the Lucene display results that match a conjugated form of the query?
faq79: How do I make the Lucene display results that match a conjugated form of the term?
faq79: How do I make the Lucene display that match a conjugated form of the word?
faq79: How do I make the Lucene display results that match a root form of the query?
faq79: How do I make the Lucene display results that match a root form of the term?
faq79: How do I make the Lucene display that match a root form of the word?
faq79: What is the PorterStemFilter?
faq80: Does Lucene search for aliases forms of the term?
faq80: Does Lucene search for alias forms of the term?
faq80: How can I make the Lucene search for terms related to the query term?
faq80: How can I make the Lucene return terms related to the query term?
faq80: How can I make the results show documents that match related terms?
faq80: How can I make the results display documents that match related terms?
faq81: How are the hits ranked?
faq81: How are the results ranked?
faq81: How are the matches scored?
faq81: How are the matches ranked?
faq81: How are scores assigned to hits?
faq81: How are scores assigned to results?
faq81: How are scores assigned to matches?
faq81: How are scores assigned to ranked?
faq81: What is the Lucene's scoring algorithm?
faq81: What is the Lucene's ranking algorithm?
faq82: How does the position of the matches in the text affect the scoring?
faq82: How does the position of the matches in the text affect the ranking?

faq82: How does the position of the matches in the field affect the scoring?
faq82: How does the position of the matches in the field affect the ranking?
faq82: Does the position of the matches in the text have any effect on the scoring?
faq82: Does the position of the matches in the text have any effect on the ranking?
faq82: Does the position of the matches in the field have any effect on the scoring?
faq82: Does the position of the matches in the field have any effect on the ranking?
faq83: How does the length of a field affect the ranking of the matches?
faq83: How does the length of a field affect the scoring of the matches?
faq83: How does the length of a field affect the ranking of the hits?
faq83: How does the length of a field affect the scoring of the hits?
faq83: Does the length of a field have any effect on the ranking of the matches?
faq83: Does the length of a field have any effect on the ranking of the hits?
faq83: Does the length of a field have any effect on the ranking of the results?
faq83: Does the length of a field have any effect on the scoring of the matches?
faq83: Does the length of a field have any effect on the scoring of the hits?
faq83: Does the length of a field have any effect on the scoring of the results?
faq83: Is the ranking higher when the query matches a term in a shorter word field?
faq83: Is the scoring higher when the query matches a term in a shorter word field?
faq83: Is the ranking higher when the query matches a term in a short word field?
faq83: Is the scoring higher when the query matches a term in a short word field?
faq83: Is the ranking lower when the query matches a term in a longer word field?
faq83: Is the scoring lower when the query matches a term in a longer word field?
faq83: Is the ranking lower when the query matches a term in a long word field?
faq83: Is the scoring lower when the query matches a term in a long word field?
faq84: How do I increase the score of certain document types?
faq84: How do I increase the score of particular document types?
faq84: How do I increase the score of specific document types?
faq84: How do I increase the score of various document types?
faq84: How do I increase the ranking of certain document types?
faq84: How do I increase the ranking of particular document types?
faq84: How do I increase the ranking of specific document types?
faq84: How do I increase the ranking of various document types?
faq84: How do I boost the score of certain document types?
faq84: How do I boost the score of particular document types?
faq84: How do I boost the score of specific document types?
faq84: How do I boost the score of various document types?
faq84: How do I increase the rank of certain document types?
faq84: How do I increase the rank of particular document types?
faq84: How do I increase the rank of specific document types?
faq84: How do I increase the rank of various document types?
faq85: How do I show excerpts from the hits?

faq85: How do I show excerpts from the results?
faq85: How do I show excerpts from the matches?
faq85: How do I show extracts from the hits?
faq85: How do I show extracts from the results?
faq85: How do I show extracts from the matches?
faq85: How do I show the selections from the hits?
faq85: How do I show the selections from the results?
faq85: How do I show the selections from the matches?
faq85: How do I the highlight the matched words?
faq85: Does Lucene provide the functionality to display excerpts or highlight excerpts?
faq86: How can I make the results show a cached version of the matches with the matched words highlighted?
faq86: How can I make the results display a cached version of the matches with the matched words highlighted?
faq86: How can I make the results return a cached version of the matches with the matched words highlighted?
faq86: How can I make the results show a cached version of the hits with the matched words highlighted?
faq86: How can I make the results display a cached version of the hits with the matched words highlighted?
faq86: How can I make the results return a cached version of the hits with the matched words highlighted?
faq87: How can I perform a search on a specific group of documents?
faq87: What are the ways in which I can restrict the search to a subset of documents?
faq87: What are the ways in which I can limit the search to a subset of documents?
faq87: What are the ways in which I can restrict the search to a group of documents?
faq87: What are the ways in which I can limit the search to a group of documents?
faq87: How can I make the results display only the matches in a subset of documents?
faq87: How can I make the hits display only the matches in a subset of documents?
faq87: How can I make the results display only the matches in a group of documents?
faq87: How can I make the hits display only the matches in a group of documents?
faq87: How do I use multi indices to search a subset of documents?
faq87: How do I use Document Type Field to search a subset of documents?
faq88: How do I perform a search limited to a subtree?
faq88: How do I perform a search limited to a subdirectory?
faq88: How do I perform a search limited to a subfolders?
faq88: How do I perform a search restricted to a subtree?
faq88: How do I perform a search restricted to a subdirectory?
faq88: How do I perform a search restricted to subfolders?
faq88: How can I make Lucene return only results in a particular subtree?
faq88: How can I make Lucene return only results in a particular subdirectory?

faq88: How can I make Lucene return only results in particular subfolders?
faq88: How can I make Lucene return only results in a specific subtree?
faq88: How can I make Lucene return only results in a specific subdirectory?
faq88: How can I make Lucene return only results in specific subfolders?
faq88: How can I make Lucene return only results in a certain subfolders?
faq88: How can I make Lucene return only results in a certain subdirectory?
faq88: How can I make Lucene return only results in a certain subtree?
faq88: How can I make Lucene return only hits in a particular subtree?
faq88: How can I make Lucene return only hits in a particular subdirectory?
faq88: How can I make Lucene return only hits in particular subfolders?
faq88: How can I make Lucene return only hits in a specific subtree?
faq88: How can I make Lucene return only hits in a specific subdirectory?
faq88: How can I make Lucene return only hits in specific subfolders?
faq88: How can I make Lucene return only hits in a certain subfolders?
faq88: How can I make Lucene return only hits in a certain subdirectory?
faq88: How can I make Lucene return only hits in a certain subtree?
faq88: How can I make Lucene return only matches in a particular subtree?
faq88: How can I make Lucene return only matches in a particular subdirectory?
faq88: How can I make Lucene return only matches in particular subfolders?
faq88: How can I make Lucene return only matches in a specific subtree?
faq88: How can I make Lucene return only matches in a specific subdirectory?
faq88: How can I make Lucene return only matches in specific subfolders?
faq88: How can I make Lucene return only matches in a certain subfolders?
faq88: How can I make Lucene return only matches in a certain subdirectory?
faq88: How can I make Lucene return only matches in a certain subtree?
faq89: How can I exclude a subtree in searches?
faq89: How can I exclude a subdirectory in searches?
faq89: How can I exclude a subfolder in searches?
faq89: How can I erase a subtree from a hierarchical index?
faq89: How can I erase a subdirectory from a hierarchical index?
faq89: How can I erase a subfolder from a hierarchical index?
faq89: How can I exclude a subtree from a hierarchical index?
faq89: How can I exclude a subdirectory from a hierarchical index?
faq89: How can I exclude a subfolder from a hierarchical index?
faq89: How can I delete a subtree from a hierarchical index?
faq89: How can I delete a subdirectory from a hierarchical index?
faq89: How can I delete a subfolder from a hierarchical index?
faq89: How can I make a certain subdirectory inaccessible invisible to the user?
faq89: How can I make a particular subtree inaccessible to the user?
faq89: How can I make a specific subfolder invisible to the user?
faq90: How can I reduce the expense of trying to read large documents from the index?

faq90: How can I prevent Lucene from reading large documents from the index?
faq90: How can I perform searches that do not require reading documents from the index?
faq90: What is the best way to perform searches with large documents?
faq91: Can I modify the index and perform searches at the same time?
faq91: Can I edit the index and perform searches at the same time?
faq91: Can I modify the index and search concurrently?
faq91: Can I edit the index and search concurrently?
faq92: Can I prevent an unauthorized user from accessing secure documents?
faq92: How do I eliminate the result that do not match a security criteria?
faq92: How do I filter the result that does not match security criteria?
faq92: How do I hide the result that does not match security criteria?
faq92: How do I eliminate the hits that do not match security criteria?
faq92: How do I filter the hits that do not match security criteria?
faq92: How do I hide the hits that do not match security criteria?
faq92: How do I eliminate the result of secure documents?
faq92: How do I filter the result of secure documents?
faq92: How do I hide the result of secure documents?
faq92: How do I eliminate the hits of secure documents?
faq92: How do I filter the hits of secure documents?
faq92: How do I hide the hits of secure documents?
faq92: How do I hide hits for unauthorized users?
faq92: How do I eliminate results for unauthorized users?
faq92: How to I prevent the search engine from returning results to unauthorized users?
faq92: How do I keep secure documents from being visible to a user?
faq92: How do I prevent secure documents from being visible to a user?
faq92: How can I limit the results displayed for security reasons?
faq92: How can I restrict the results displayed for security reasons?
faq92: How do I make certain documents inaccessible to unauthorized users?
faq92: How do I make certain documents invisible to unauthorized users?
faq93: How can I submit a question for this FAQ?
faq93: How can I suggest a question for this FAQ?
faq93: How can I contact the moderator of this FAQ?
faq93: How can I contact the administrator of this FAQ?
faq93: How can I contact the maintainer of this FAQ?
faq94: Does this FAQ need volunteers?
faq94: Does this FAQ accept volunteers?
faq94: Can I help maintain this FAQ?
faq94: Can I help update this FAQ?
faq94: Can I help edit this FAQ?
faq94: Can I help review this FAQ?
faq94: How can I contribute to this FAQ?

faq95: What are the licensing terms of this FAQ?
faq95: What is the copyright information regarding this FAQ?

Reference

[BAS94] Buckley, C., Allan, J., Salton, G. Automatic routing and ad-hoc retrieval using SMART: TREC 2, The Second Text REtrieval Conference (TREC-2), NIST Special Publication 500-215, National Institute of Standards and Technology, Gaithersburg, MD, pp. 45-55, March 1994.

[BCB92] Bartell, B.T., Cottrell, G.W., Belew, R.K. Latent Semantic Indexing is an optimal special case of Multidimensional Scaling, Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 161-167, 1992.

[BHK+97] R. Burke, K. Hammond, V. Kulyukin, S. Lytinen, N. Tomuro, and S. Schoenberg, "Natural Language Processing in the FAQ Finder System: Results and Prospects", in Working Notes from AAAI Spring Symposium on NLP on the WWW, 1997, pp. 17-26

[BK811] D.A. Buell, D.H. Kraft, 1981 A model for a weighted retrieval system, J. of ASIS, pp.211-216.

[BOO80] Bookstein A. Fuzzy requests: an approach to weighted Boolean searchs. Journal of the American Society for Information Science 1980; 31(4):240-247

[Buc85] Buckley, C. 1985. Implementation of the SMART Information Retrieval Retrieval [sic] System. Technical Report 85-686, Cornell University

[Bue81] Buell DA. A general model of query processing in information retrieval system. Information Processing &Management 1981:17(5)249-262

[COO88] Cooper, W. (1988). Getting Beyond Boole. Information Processing & Management, 24, 243-24

[Cro87] W. Bruce Croft, Approaches to intelligent information retrieval, Information Processing and Management: an International Journal, v.23 n.4, p.249-254, July 1987

[CT03] L. Chen and N. Tokuda, Bug Diagnosis By String Matching: Application to ILTS for Translation, CALICO Journal, 20(2003), No. 2, 227 — 244

[CTN01] L. Chen, N. Tokuda and A. Nagai, "Probabilistic Information Retrieval Method Based on Differential Latent Semantic Index Space", IEICE Trans. On Information and Systems, E84-D (2001), no. 7, 910—914.

[CTN03] L. Chen, N. Tokuda, and A. Nagai. A new differential LSI space-based probabilistic document classifier. Information Processing Letters, 88:203–212, 2003.

[CTN04] L. Chen, N. Tokuda, and A. Nagai. A Combined Concept- and Syntactic-Based FAQ System, manuscript, Feb., 2004

[CWN+02] Cui,H.,Wen,J.-R.,Nie,J.-Y.,and Ma, W.-Y. Probabilistic query expansion using query logs. In Proceedings of the eleventh international conference on World Wide Web (2002), ACM Press, pp. 325—332

[Dum92] S. Dumais, Enhancing Performance in Latent Semantic Indexing (LSI) Retrieval, Technical Report, Bellcore (now Telcordia Technologies), Morristown, NJ (September 1992).

[DDF+90] Deerwester, S., Dumais, S., Furnas, G., Landauer, T., Harshman, R. Indexing by latent semantic analysis, Journal of the American Society for Information Science, 41(6), pp. 391-407, 1990.

[Eft95] Efthimiadis, E.N. User Choices: A new yardstick for the evaluation of ranking algorithms for interactive query expansion, *Information Processing & Management,* Vol. 31, No. 4, pp. 605-620, 1995.

[Mil90] Miller, G.A. WordNet: An on-line lexical database, International Journal of Lexicography, 3(4), 1990.

[FLG+87] Furnas, G.W., Landauer, T.K., Gomez, L.M. and Dumais, S.T. 1987. The vocabulary problem in human-system communication. Commun. ACM 30, 11 (Nov. 1987), Pages 964-971

[Fra92] William B. Frakes, Information retrieval: Data Structures and Algorithms Pages: 131- 160, Prentice-Hall, Inc, Year of Publication:1992

[Har92] Harman, D. Relevance feedback revisited, Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp.1-10, 1992.

[HPP+95] Hearst, M., Pedersen, J., Pirolli, P. Schutze, H. Xerox TREC4 Site Report, Text REtrieval Conference-4, Gaithersburg, MD, National Institute of Standards and Technology, November 1-3, 1995.

[Hul94] Hull, D. Improving text retrieval for the routing problem using Latent Semantic Indexing, Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp.282-291, 1994.

[JXu97] Xu, J. Solving the word mismatch problem through automatic text analysis. Ph.D. Thesis, Department of Computer Science, University of Massachusetts, Amherst, MA, USA, May 1997. 21

[Lee94] Lee, J.H. Properties of extended Boolean models in information retrieval, Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 182-190, 1994.

[LKL92] Lee J.H., Kim M.H., Lee Y.J.. Enhancing the fuzzy set model for high quality document rankings. In: Processing of the 19th Euromicro Conference 1992, pp.337-344.

[LKK+93] Lee J.H., Kim W.Y., Kim M.H., Lee Y.J.. On the evaluation of Boolean operators in the extended Boolean retrieval framework. In: Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. 1993 pp.291-297

[Por80] M. Porter. An algorithm for suffix stripping. Program,14(3):130-137, 1980.

[RB99] R. Baeza-Yates and B. Ribeiro-Neto. Modern Information Retrieval. Addison-Wesley, Wokingham, UK, 1999.

[Sal73] Salton G., Experiments in Multi-Lingual Information Retrieval. Information Processing Letters, 2(1), 6-11. 1973

[Sal88] Salton, Gerard. Automatic Text Processing. Addison-Wesley Publishing Company, 1988

[Sal89] Salton, G. Automatic text processing: The transformation, analysis, and retrieval of information by computer, Addison-Wesley, Reading, MA, 1989.

[SB90] Salton, G., Buckley, C. Improving retrieval performance by relevance feedback, *Journal of the American Society for Information Science*, 41(4), pp. 288-297, 1990.

[SMM83] Salton, G., McGill, M.J. Introduction to Modern Information Retrieval, McGraw Hill Publishing Company, New York, 1983.

[SFW83] Salton G, Fox EA, Wu H. Extended Boolean Information Retrieval. Communications of the ACM 1983; 26(11): 1022-1036

[Sne 99] E. Sneiders. Automated FAQ Answering: Continued Experience with Shallow Language Understanding. In Question Answering Systems. Papers from the 1999 AAAI Fall Symposium. Technical Report

FS-99-02, 1999

[SS97] Hinrich Schütze, Craig Silverstein, Projections for efficient document clustering, Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval, p.74-81, July 27-31, 1997, Philadelphia, Pennsylvania, United States

[TC90] H. Turtle , W. B. Croft, Inference networks for document retrieval, Proceedings of the 13th annual international ACM SIGIR conference on Research and development in information retrieval, p.1-24, September 05-07, 1990, Brussels, Belgium

[vR79] van Rijsbergen, C. J. Information retrieval. Butterworths, 1979

[VVW+97] Venkat N. Gudivada, Vijay V. Raghavan, William I. Grosky, Rajesh Kasanagottu, Information Retrieval on World Wide Web, IEEE Computer Society, September-October 1997 (Vol. 1, No. 5) p p. 58-68.

[Whi95] Whitehead, S. D. 1995. Auto-FAQ: an Experiment in Cyberspace Leveraging. Computer Networks and ISDN Systems, vol. 28, no. 1-2: 137-146.

[XC98] Xu, J., Callan, J. Effective Retrieval with Distributed Collections, Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 112-120, 1998.